Vela® P R I S M
MPEG-2 Transcoder

API Developer's Guide
Version 2.6.5

Application Programming Interface Documentation for the Vela CineView Pro Prism MPEG-2 Transcoder

Vela's Real-Time MPEG-2 Transcoding Solution

Document Part Number 9050-2100LRW

5733 Myerlake Circle
Clearwater, FL 33760-2804
Phone: (727) 507-5300
Fax: (727) 573-5310
World Wide Web – http://www.vela.com

Mailing / Shipping Address:
5733 Myerlake Circle
Clearwater. FL 33760-2804

All returns must be accompanied by an authorized RMA number obtained from Vela.

Published in the United States of America 6-03 Rev. PAPI-0306-003

Table of Contents

List of Figures and Tables

Chapter 1

Getting Started

Introduction to the Vela Prism API

This Application Programming Interface (API) for the Vela CineView® Pro Prism MPEG-2 transcoder was designed using an object-oriented approach. Each core function of the transcoder has its own COM (Microsoft's Component Object Model) component associated with it. A complete transcode is accomplished when these components are used together and accessed through Filter Manager, Vela's single, well-defined COM interface. The Filter Manager interface is responsible for managing a transcoding session by reading Windows® Registry settings of the storage of the last byte of transcoded material. As a developer, you only need to Initialize, Cue, Start, Stop, Pause, and Resume. The Filter Manager handles the rest.

If you have already developed a software application for the standard Argus® encoder, the good news is that you won't need to do much programming to turn on multi-stream transcoding. In fact, all that you really need to do is to make some adjustments to the Windows® Registry, toggling a flag to turn on each of the output streams, then setting the transcoding parameters for that stream. In Appendix A you will find a detailed description of each of the Registry tables that are used specifically to configure the Vela CineView Pro decoder for transcoding.

You may find it helpful to look over our encoder user guide, the *Argus Spectrum Ver. 2.6 Installation and User Manual,* which describes the operation of the Argus Spectrum multi-stream encoder. Among other topics, it includes a discussion of the Aladdin HASP™ software protection key (sometimes know as a “dongle”), that grants or denies permission to access and use each of the multi-stream encoding components. HASP dongles are used with Vela CineView Pro Prism components.

As we mentioned earlier, you have the option of turning on one, two, or all three of the optional streams, assuming that the HASP device attached to your PC grants permission to use them. You should keep in mind, though, that each transcoder requires additional processing power. You may find that running all three of the low-bitrate transcodes at once pushes your CPU usage prohibitively high. As you experiment with transcoding, remember that you can reduce the

CPU usage of any one of the low-bitrate transcodes by decreasing its horizontal and vertical resolutions and by decreasing its bit rate. Observing the Windows task manager as you transcode should help you to determine the ideal settings for your customized Vela Prism transcoder.

Minimum System Requirements

- Microsoft Windows 2000 (Service Pack 2) or Windows NT 4.0 operating system (Service Pack 6a).
- IBM® PC or PC-compatible Pentium® III dual-processor (866 Mhz each) system with PCI bus
- 256 MB RAM
- CD-ROM drive (for installation of system files)
- Vela CineView Pro PCI/VGA decoder board
- 9.0 GB hard drive (fast/wide SCSI)
- Aladdin HASP™ software protection key ("dongle"), programmed to enable the transcoding options that you have purchased. This device is supplied when you purchase the Vela Prism transcoder or Argus Spectrum encoder.

Software Requirements

- Vela system software, version 2.6.5
- Compiler with COM support: Microsoft Visual C++™ 6.0 (Service Pack 4 or higher) with Unicode support; or Visual Basic™ 6.

Included Files

The following table is a list of all files to be installed with the CineView Pro Prism and with the installation of the Prism SDK. (Those files that are installed as part of the SDK are located in the folder C:\Program Files\Vela Research\CineView Pro\Prism\SDK or in one of its sub-folders.)

Vela Prism SDK Included Files		
Filename	**File Description**	**Folder**
CinProSerComU.dll FilterManagerU.dll MultiplexU.dll RemoteStoreU.dll	Various COM components.	C:\Program Files\Vela Research\Cineview Pro\Prism
AsfEncodeU.dll AsfWriterU.dll	DLLs for Windows Media Format transcodes.	C:\Program Files\Vela Research\Common\WMF
LigosEncodeU.dll	DLL for Ligos MPEG-1 transcodes.	C:\Program Files\Vela Research\Common\MPEG
RealEncode.dll	DLL for Real (G2) transcodes.	C:\Program Files\Vela Research\Common\G2
MemoryManager.dll MemMgrServer.exe Vela_Pins.dll	Server executables required to communicate with CineView Pro decoder.	C:\Program Files\Vela Research\Common
eula.txt	End-user license agreement for EDL Editor.	C:\Program Files\Vela Research\CineView Pro\Prism
FilterManager.tlb	Type library for Filter Manager COM component.	C:\Program Files\Vela Research\CineView Pro\Prism\SDK\Include
TranscodeClient.DSP & associated source code	Work space containing source code for the C++ sample application that drives the transcoder and sets the Registry.	C:\Program Files\Vela Research\Cineview Pro\Prism\SDK
CVProPrism.exe	Build of TranscodeClient work space. Allows user to test sample application without building it.	C:\Program Files\Vela Research\Cineview Pro\Prism

Table 1-1. Vela Prism SDK Included Files

Vela Prism SDK Included Files (Continued)		
Filename	**File Description**	**Folder**
Wininet	System DLL required to support RemoteStore component.	C:\WINNT\System32
Various	System DLLs for ATL and MFC.	C:\WINNT\System32

Table 1-1. Vela Prism SDK Included Files (Continued)

Component Summary

The goal of this API set is to give you, the developer, a binary-independent, easy-to-control interface to the Vela Prism transcoder. We have implemented the transcoder software on the Windows platform as a set of COM components. You are required to create only a single COM object, one derived from the Filter Manager interface. Filter Manager, in turn, controls all of the subordinate COM components for you.

Filter Manager

The Filter Manager interface is easy to use, exposing just a handful of core methods or commands: Initialize, Load, Cue, Start, Stop, Pause, Resume, and Reset. Each of these basic commands, explained in detail later in this manual, triggers the start of a specific phase of the transcoding process.

To establish the parameters under which the transcode is to operate, you will set Prism-specific Windows Registry keys, grouped by core functionality in tables. For example, to configure the transcoder to perform a Ligos transcode, you would set the keys in the DualEnc and the LigosMpeg1 Registry tables. Each of the Registry tables is explained in detail in Appendix A.

You can set the Registry keys by programmatically using the CRegistry class (or your own software) to write to the Windows Registry. This method is useful for changing the values of keys that must be reset with each transcoding session (the name of the output file, for example).

The COM components that drive the CineView Pro Prism transcoder are all self-registering. The installation program registers each component using a utility, REGSVR32.EXE, that is included with the Prism API. This same utility can be used to remove components from the Registry, to add new components, or to replace existing components with newer versions.

Vela Prism SDK Installation

Note that if any previous version of Vela Prism software is installed on your system, it **must** be uninstalled before you continue with the installation of the version 2.6.5 SDK. Use the Windows Control Panel "Add/Remove Programs" function to uninstall Prism software, if necessary.

After you click the "Install CineView Pro/LE/XL" or "Install CineView Pro Prism" option from the Vela system software Auto-Run screen (Figure 1-1), the "Select Components" screen will appear (Figure 1-2). From this screen you can select, among other options, which of the transcode modules you wish to install. You can also choose the particular CineView Pro SDK you wish to install, including the Prism SDK. Simply click on the checkbox(es) of the item(s) you want.

Note that the installation of the SDK is a **password-protected** process. Included with the SDK is an authenticated password that allows installation of the SDK and accompanying files. If you did not receive a password with your CD-ROM, contact Vela Support.

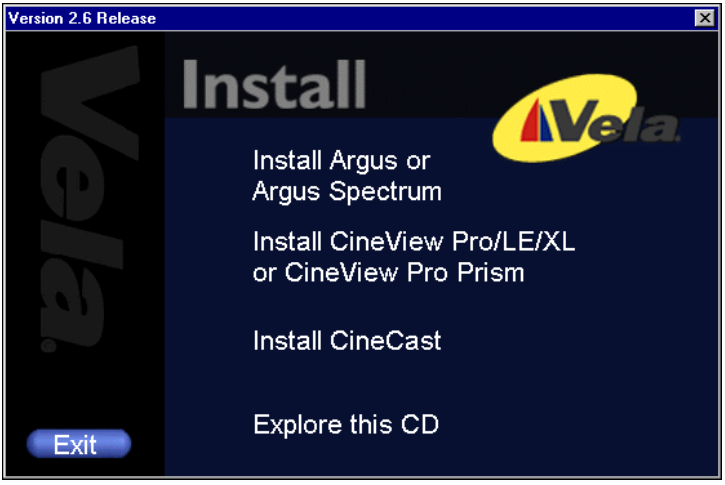

Figure 1-1. Installation Auto-Run Screen

Suggested Reading

This manual assumes that you are familiar with ATL, COM, and C++ (or Visual Basic) programming. For more information on these topics, we recommend that you refer to the following publications.

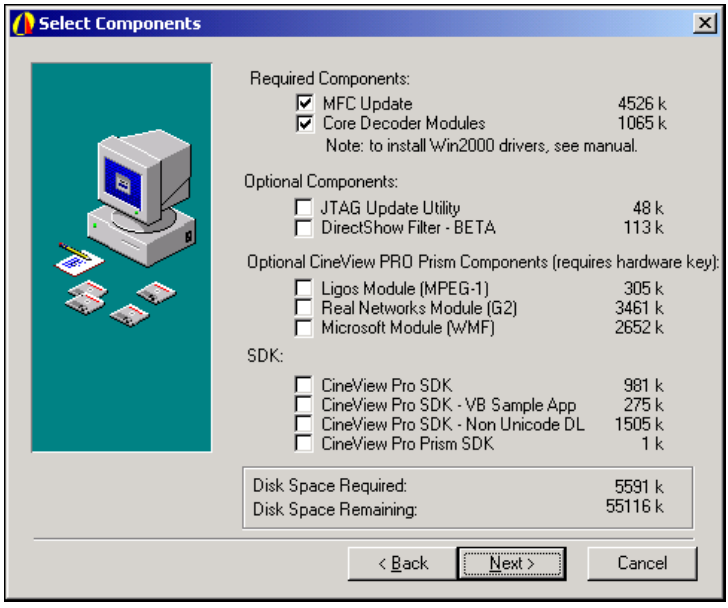

Figure 1-2. Select Components Screen

ATL/COM References

Inside COM, Rogerson; Microsoft Press. Recommended for COM introduction. Covers COM programming explicitly from a C/C++ hard-core, low-level mode.

ATL Internals, Rector and Sells; Addison-Wesley. This is an excellent reference for ATL programming using Visual Studio 6.0. It includes very useful sections on Smart Pointers, BSTRs, and events.

ATL COM, Grimes, Stockton, Reilly, and Templeton; WROX Press. This book delves deep into the heart of the Active Template Library. Primarily deals with server-side issues, but has some client code development considerations as well.

C++ References

The C++ Programming Language, Stroustrup. This bottom line reference on the C++ programming language is highly recommended for the serious developer.

Using Visual C++, Gregory; QUE Publishing. A comprehensive reference for Microsoft's VC++ compiler.

Customer Support

In the event of questions or problems with Vela Application Programming Interface methods, materials, or this manual, do not hesitate to contact Vela Training and Support as follows:

- Phone: (727) 507-5301
- E-mail: support@vela.com
- World Wide Web - http://www.vela.com

Chapter 2

Using the Filter Manager API

Component Overview

The key element of the Vela CineView Pro Prism API is the **Filter Manager** COM component, which offers two custom interfaces. The primary interface allows your application to make requests of the Filter Manager. The second custom interface (the outgoing interface) allows the Filter Manager to send COM events to your calling application. Note that both Filter Manager interfaces use Unicode-style character strings.

The Primary Interface

The primary Filter Manager interface exposes methods that service requests for transcoder functionality. Specifically, it accepts requests to initialize and reset the transcoder software, as well as requests to cue, start, and stop a transcoding session.

Through its primary interface, the Filter Manager component exposes *methods* and *properties*. If you are unfamiliar with methods and properties, or with other aspects of object-oriented programming, take time to review reading material on C++ or COM. Refer to the end of Chapter 1 for some suggestions.

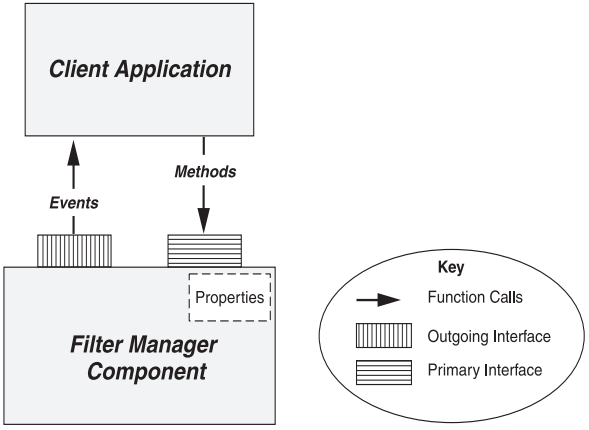

Figure 2-1. Filter Manager Interfaces

A *method* is simply a function call. Usually this function performs an operation, then returns a status code. Each of the fully supported Filter Manager methods is defined in this manual. The definition includes a description of the operation

that the method performs as well as a list of the possible return values. Be sure to check the return value of any method that you call before proceeding with the transcoding process.

Similar to a C++ class data member, a *property* has a value or a setting. Typically the value of a property can be set by calling a specific Put() method exposed by the Filter Manager interface. Likewise, its value can be retrieved with a Get() method. We concentrate more on the Windows Registry and less on COM properties to set and retrieve transcoding parameters. In fact, Filter Manager exposes just two supported properties (see "Filter Manager Interface Properties," page 13).

The Secondary (Outgoing) Interface

Through its outgoing interface, the Filter Manager component implements *events*. An event is a COM mechanism that allows the component to send messages to the calling application. Filter Manager uses events to issue Log messages, Error messages, Pause/Resume messages, and Finished messages to the client application. The client can register to receive these messages, at your discretion.

The remainder of this chapter describes techniques of setting transcoding parameters for the CineView Pro Prism. Additionally, it defines and describes each of the basic transcoding commands exposed through the primary Filter Manager interface.

System Configuration Settings

During the initialization process, the Filter Manager interface checks the Windows Registry to determine if the CineViewPro decoder is loaded.

The Windows Registry table HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\EncoderConfig has one key: "DecoderInstalled."

NOTE: This key must have the value of "1."

Transcode Parameters: The Windows Registry

The CineView Pro Prism transcoder software uses the Windows Registry exclusively to store and retrieve parameters for a transcoding session.

Figure 2-2 illustrates typical interactions between Prism-related software and the Windows Registry.

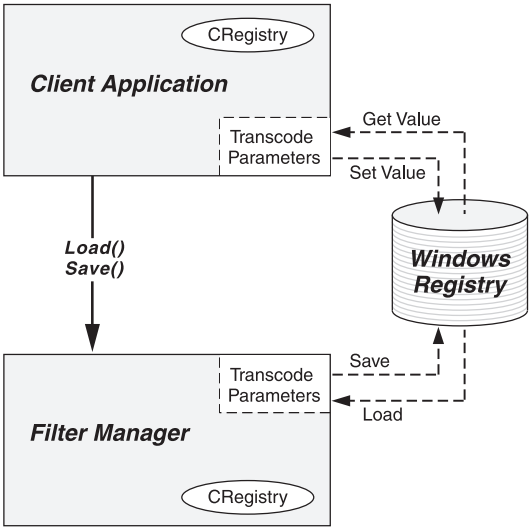

Figure 2-2. Windows Registry Transactions

One useful feature of the Registry method of storing transcoding parameters is that it is not necessary to set up the Registry before the first transcode. If the application attempts to load from the Registry when there are no transcode settings there, Filter Manager responds by saving all of the default settings to the Registry, creating all of the needed keys. You can then programmatically or manually (using Microsoft's regedt32 or regedit tool) change Registry settings before subsequent transcodes.

All of the Registry settings for the CineView Pro Prism are stored in one of seven Registry locations under HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus. These sub-locations are: “EncoderConfig,” “FilterMgr,” “DualEnc,” “LigosMpeg1,” “LigosMux,” “RealNetworks,” and “Wmf.” Appendix A identifies and describes in detail each of the Registry settings that support the CineView Pro Prism transcode process.

There are a number of circumstances under which you may need to access the transcoding parameters stored in the Windows Registry. As shown in Table 2-1, the Prism SDK provides a variety of tools to assist with the task of managing these transcoding parameters.

Task	Tool	Description
	CVProPrism	An application that displays all of the transcoding parameters, allowing the user to review and modify them. It also has the capability to start and stop transcodes. Source code is provided in Chapter 3.
Change individual Registry settings (i.e., file name, mark-in) before starting a transcode.	CRegistry class, SetValue() method	The CRegistry class, whose source code is provided with the CVProPrism application, provides easy-to-use GetValue() and SetValue() commands to manage transcoding parameters of all data types.
Display a specific set of transcode parameters through the user interface to the transcoder.	CRegistry class, GetValue() method	
Load the full set of transcoding parameters from the Windows Registry in preparation for a transcode.	FilterManager Load() method	Loads all of the transcode parameters from the Registry into the specific transcoder COM components to which the parameters apply.
Save the full set of transcoding parameters under which Prism is currently transcoding.	FilterManager Save() method	Dumps all of the transcoding parameters currently in memory out to the Windows Registry.

Table 2-1. Managing Transcode Parameters

Changing Individual Registry Settings

The production release of the CineView Pro Prism SDK, version 2.6.5, contains source code for a sample application that illustrates the loading, modification, and saving of each of the individual Registry settings. This source code is installed in the C:\Program Files\Vela Research\CineView Pro\Prism\SDK folder. You'll probably find that you can set most of the transcoding properties once using a Registry editing tool like the CVProPrism application, then leave the settings untouched.

However, a few properties (for example, the input file name and the output file name) are likely to change with each transcode. Using the source code of the CVProPrism application as an example, you can programmatically change these settings before loading and cueing for each transcoding session.

The CVProPrism application includes source code for a Registry-management class, CRegistry, that makes it easy for you to access and change Registry settings.

Registry-Access Methods Exposed Through Filter Manager

The Filter Manager interface exposes a Load() method that loads the full set of transcode parameters from the Registry into memory, as well as a Save() method that writes all of Prism's current property settings to the Registry. Before calling the FilterManager Cue() method to set up for a transcode, you should first write to the Registry any individual property changes that you need to make, then call Load() to load all of the Prism transcoder settings into memory. Refer to the source code of our sample application, CVProPrism, for an example. You can preserve the current Prism settings by calling Save() with each transcode.

long Load() – Loads all of the settings from the Registry locations listed above, except for those in the EncoderConfig table (this table is read only once, during the Filter Manager initialization process). If the Registry does not yet exist, the Load() call creates it, enters all of the Registry keys, and assigns their default values.

The Load() method returns 0 if it successfully loads transcoding parameters from the Registry. If it fails, it returns a negative number (defined in Appendix B).

long Save() – Saves to the appropriate Registry keys all of the transcoding parameters that are currently in memory, except for those in the EncoderConfig table. It returns 0 if the save procedure finished successfully or a negative error code (defined in Appendix B) if the procedure was unsuccessful.

Filter Manager Interface Properties

Because the Windows Registry is used to store all of the transcoding parameters, the list of Filter Manager interface properties defined in this section is short. In fact, it includes only those properties that report on the version numbers of installed hardware and firmware, those that report on the progress or status of a transcoding session, and those that toggle on and off specific Windows functionality.

Listed below are the Filter Manager interface properties that are fully supported in version 2.6.5. Though a number of previously used properties may still appear in the Filter Manager interface, you should assume that properties not defined below are not fully supported.

Before defining the property, each listing specifies whether the Get() method or the Put() method is implemented for that property. In each of the definitions, the data-type of the property is listed within the set of parentheses. In cases where the Get() method is implemented, this data type specifies the return value of the Get() method. In cases where the Put() method is implemented, the data type within the parentheses indicates the data type of the setting that is to be passed as an argument.

For example, if a property is listed as **PropertyX(long)**, then assume that the definition of the corresponding Put() method is **void PutPropertyX(long val)**, and that the definition of the corresponding Get() method is **long GetPropertyX()**;.

Transcode (BOOL): *(Put)* If this property is set to 1, the Filter Manager knows that transcoding is enabled and to load the necessary components. When the property is set to 0, transcoding will not function. Note that this method must be called before any other Filter Manager method is called, including Initialize().

WriteToMessageBox(BOOL): *(Put)* If this property is set to 1, the Filter Manager software will pop up Windows message boxes when specific errors occur. When the property is set to 0, those message boxes will be suppressed. This method should be called before any other Filter Manager method is called.

Unsupported Properties

Interface properties not described in the above sections are not currently supported in this version.

Basic Filter Manager Methods

In addition to methods designed to load and save transcode parameters, the Filter Manager interface offers eight basic methods that control the operation of the Prism transcoder. Each of these methods sets the value of *result* to report its success or failure in carrying out the requested task. Generally speaking, if *result* is 0 or positive when the method returns, then the method ended successfully. Negative values represent failure. The allowable state transitions for the CineView Pro Prism transcoder are depicted in the table on the following page:

Allowable State Transitions			
Current State	**Allowed Commands**	**Resulting State**	
		Success	**Failure**
Start State	Initialize()	Initialized	Exit
	Exit		
Initialized	Reset()	Reset state	Exit
	Exit		

Table 2-2. Prism Spectrum Allowable State Transitions

Allowable State Transitions (Continued)			
Current State	**Allowed Commands**	**Resulting State**	
		Success	**Failure**
Reset State	Cue()	Cued	Initialized
Cued	Start()	Started	Initialized
Started	Pause()	Paused	Initialized
	End() or Stop()	Initialized	Initialized
Paused	Resume()	Resumed	Initialized
Resumed	Pause()	Paused	Initialized
	End() or Stop()	Initialized	Initialized

Table 2-2. Prism Spectrum Allowable State Transitions (Continued)

long Initialize() – Sets up the transcoder application, creating an instance of each required COM object. The Initialize() method should be called only once during the life of a single application. It must be called **after** the PutTranscode() method and the PutWriteToMessageBox() method. Initialize() returns 0 if successful. Otherwise, it returns one of the error codes listed in Appendix B, "Filter Manager Error/Status Codes."

long Cue() – Should be called **after** a call to Load(), which loads Registry settings. Cue() sets up each component for a transcode, based on the Registry settings that apply. The Cue() method returns 0 if it is successful. Otherwise, it returns one of the error codes listed in Appendix B, "Filter Manager Error/ Status Codes."

long Start() – This method starts the transcode immediately. Returns 0 if successful, or one of the error codes listed in Appendix B if not.

long Stop() – Stops the transcode. Actually, this is one of two methods of stopping a transcode:

1. Calling the Stop() method causes the transcode to stop immediately; it is actually a call to abort the transcoding process. The Stop() method returns 0 if it is successful. If not, it returns one of the error codes listed in Appendix B.

2. The End() call requests that the transcode terminate cleanly after completely processing the current frame in all of the components.

long End() – This is an alternate and preferred method of stopping the transcode. It performs a "clean" stop, guaranteeing that the frame currently being processed by the components will be flushed through all transcode modules before the transcode halts. See the discussion of stopping strategies, directly above. The End() method returns a 0 if it is successful. It returns an error code (listed in Appendix B) if it encounters an error.

long Reset(). Causes all of the Prism's COM components to reset themselves in preparation for the next transcode. The Reset() method should be called prior to each Cue call. The Reset() method returns a 0 if it is successful, or an error code listed in Appendix B if not.

Events

The Filter Manager uses the COM event mechanism to send messages to its client through the outgoing interface. An *event* is a callback issued by Filter Manager in response to a noteworthy occurrence or condition. Any client of Filter Manager can choose whether or not to register and respond to Filter Manager events. Most likely, you'll want to register to receive them. Only through events can Filter Manager let your application know that it has finished a transcode — and whether or not the transcode finished successfully.

From the Filter Manager outgoing interface, a *message* and a *code* are passed each time an event is fired. The *message* is used to display a detailed message string describing the event. In the case of an Error event, the *code* specifies the error that occurred. For events other than error events, the code parameter may be 0, indicating a successful outcome. Codes with negative values are generally error codes. All others are status codes.

The Filter Manager interface supports the following events:

HRESULT ErrorEvent(long code, BSTR message) is issued when a processing error has occurred.

HRESULT LogEvent(long code, BSTR message) is issued to inform the client of the status of the transcoding process or of a recently-called method. It is informational only. If the code value is negative, the log event can be considered a warning. A log event whose code value is 99 has a special meaning and requires special processing. Filter Manager issues log events tagged with a 99 to indicate the progress of stream indexing after a Real or WMF transcode has finished. These log messages should be filtered out to prevent the log file from growing too large.

HRESULT FinishedEvent(long code, BSTR message) is issued to inform the client that the transcode is completely finished.

For an explanation of how to register events in your C++ or Visual Basic application, see the respective sections of this manual that describe C++ and Visual Basic client applications. For a more detailed explanation of COM events, refer to the recommended COM reading references listed in Chapter 1, particularly the book *ATL Internals*.

Chapter 3

The Sample Application

Overview

Developers using these components should be familiar with Microsoft Visual C++™ 6.0. Microsoft provides several wizards and tools that make adding COM support to your C++ applications relatively straightforward. While it is possible to access and use these components from other development environments, only an example for Visual C++ 6.0 is provided in this SDK. Other packages with COM support should behave in a similar manner.

The general steps for setting up a client application are as follows:

1. Create the client project.
2. Reference the COM object library (VB only).
3. Initialize the COM libraries. (VC++ only).
4. Create an instance of the desired object. (In VC++, use the Smart Pointer to the interface).
5. Use the object.
6. Uninitialize the COM libraries when finished. (VC++ only).

The remaining sections of this chapter describe and explain a working application that controls the Vela CineView Pro Prism transcoding process. When the CineView Pro Prism SDK is installed, the source code for each of these applications can be found in C:\ Program Files\Vela Research\CineView Pro\Prism\SDK. The intent of providing source code for the sample application is to illustrate the use of various programming tools to control the transcoding process. In order to present readable, easy-to-follow code, we have intentionally kept the application simple.

TranscodeClient

A Sample Visual C++ Transcoder-Control Application

Overview

A full set of source code for a C++ interface to the CineView Pro Prism transcoder, called TranscodeClient, is provided with this SDK. This application (including source code, the Filter Manager type library, and Registry-based DLLs) is located in C:\Program Files\Vela Research\CineView Pro\Prism\SDK. It would be useful to refer to a copy of the source code as you read the section that follows. Most of the source code referenced in this document is located in TranscodeClientDlg.cpp and FilterManagerEvents.cpp.

Creating the Project

When you are creating a Microsoft Foundation Class (MFC) Application Wizard EXE project like TranscodeClient, it is very important that you select from the App Wizard window the check box that adds support for ActiveX™ controls. This inserts into the stdafx.h file the header files required to support the COM libraries. If you are adding COM support to an existing project, or if your project does not use MFC, we highly suggest studying some of the books available on MFC core details and COM specifics. We also suggest using MFC as a shared DLL from App Wizard, as all of the COM components are already using this DLL.

Initializing the COM libraries

When you create a client of a COM object, you first initialize the COM libraries by calling either the **CoInitialize**, **CoInitializeEx**, or **OleInitialize** functions. The decision about which function to call is based on several factors, the most important of which is the type of threading model you want to use with the components.

All of the Vela Prism API core components support a dual interface (i.e., any interface that inherits from *IDispatch*, which is the interface that supports OLE Automation). Using the custom interface of the component provides the client with direct table access to the functionality of the component. This is much more efficient than the *IDispatch* interface which uses the COM Automation libraries to access component functionality.

In general, if you are using versions 5.0 or later of Visual C++ or Visual Basic, it is best to use the custom interface directly. The *IDispatch* interface is provided to support VB 4.0 and earlier development, as well as to access the components from scripting languages such as VBScript and JScript.

Since we are going to use the Filter Manager custom interface, we will use **CoInitialize** or **CoInitializeEx**. To determine which function to call, we need to determine the threading model we will use. Our components require that the client use the "apartment" threading model. For apartment model threading, **CoInitialize** is sufficient. The code to initialize the components looks like this:

```
{
    HRESULT hr = CoInitialize( NULL );
    if ( FAILED(hr ) )
        return FALSE;
}
```

This piece of code would normally appear in the **InitInstance** method of the **CApp** object.

Using the #import Directive

This section describes the steps required to create a COM object using Smart Pointers and the **#import** compiler directive. The **#import** directive is a Microsoft-specific compiler directive that creates a Smart Pointer wrapper class from a type library; that class can be used to create an instance of the required COM object and to use its services. To use the import directive for an instance of the Filter Manager interface, you would insert the following code in the Stdafx.h file after the "#include <Afxwin.h>" directive.

```
#include <afxdisp.h>// for AfxOleInit
…
#include <atlbase.h>
extern CComModule _Module;
#include <atlcom.h>
#include <objbase.h>
#import "FilterManager.tlb" no_namespace named_guids
```

The CComModule class implements a COM server. This allows a client to access the module's components. When you open your application, you should call _Module.Init(NULL, AfxGetInstanceHandle()). When you close the app, call _Module.Term(). For an example, please examine the InitInstance() method in TranscodeClient.cpp.

The **#import** directive creates two header files that reconstruct the type library contents in C++ source code. In this case, the files would be named **FilterManager.tlh** and **FilterManager.tli**. The primary header file, **FilterManager.tlh**, contains a typedef macro that expands to the following format:

```
typedef com_ptr_t< com_IIID<IArgusFM, __uuidof(IArgusFM)> > IArgusFMPtr
```

The C++ template class **_com_ptr_t** used in the above typedef creates a Smart Pointer (in this case, IArgusFMPtr) that can be used to access the interface passed in as the template argument (in this case, IArgusFM).

Once the Smart Pointer interface is defined in FilterManager.tlh, we can create an instance of a Smart Pointer in our sample application. Note that the only argument to the CreateInstance() method of the Smart Pointer is the class ID of the Filter Manager component. Because we specified the named_guids modifier, the **#import** directive created the required CLSID in the header file for us, eliminating the need to call CLSIDFromProgID.

```
m_pIFilterMgr.CreateInstance ( CLSID_ArgusFM ) ;
```

Once an instance of IArgusFMPtr has been created, it can be used to access the properties and events exposed through the primary Filter Manager interface.

For a more thorough discussion of Smart Pointers, refer to *ATL Internals*, by Rector and Sells.

Creating the Object

To create the interface to the Filter Manager, first declare the following in the CTranscodeClientDlg class. The Filter Manager interface defines all the basic transcoder calls: Initialize, Cue, Start, Stop, etc. The events interface defines the callbacks through which the Filter Manager component will communicate with the sample application. The class CFilterManagerEvents will be discussed in later sections of this document.

```
IArgusFMPtr             m_pIFilterMgr;
CFilterManagerEvents    *m_pFilterManagerEvents;
```

After Initializing COM, we need to create an instance of an interface to Filter-Manager and another to the FilterManager Events. The following interface creation code is located in the OnInitDialog() function in TranscodeClientDlg.cpp.

```
HRESULT hresult;
long errorCode;

// Initialize COM.
hresult = CoInitialize(NULL);
if( FAILED(hresult))
    return FALSE;

// Create an instance of the server object
errorCode = m_pIFilterMgr.CreateInstance(CLSID_ArgusFM);
if( FAILED(errorCode))
    return FALSE;

// Create the event sink object.
m_pFilterManagerEvents = new CFilterManagerEvents();

// Advise the event source that we are connecting to it.
```

```
m_pFilterManagerEvents->EasyAdvise(m_pIFilterMgr);

////////////////////////////////////////////////////////////////////////////
// The following section of code is needed only for THIS application's event-
// handling strategy.
// This sample interface handles LOG events by writing them to a log file,
// for which it needs pFile, a pointer to a file opened for writing.
// It handles Finished events with message boxes, for which it requires pWin,
// a pointer to the calling windows class.
////////////////////////////////////////////////////////////////////////////
m_pFilterManagerEvents->pView = this;
m_pFilterManagerEvents->pFile = _tfopen( _T("C:\\vela_db\\EventLog.txt"),
_T("w+"));
if( !m_pFilterManagerEvents->pFile )
{
    return FALSE;
}

// Instruct Filter Manager NOT to write directly to message boxes.
m_pIFilterMgr->PutWriteToMessageBox(FALSE);

// Transcode is enabled. If this is false then transcode will NOT work.
m_pIFilterMgr->PutTranscode(TRUE);

errorCode = m_pIFilterMgr->Initialize();
if( errorCode != 0 )
{
    exit(1);
}
```

Using the Object

Once the IArgusFMPtr is created, we can use that pointer to call any of the methods the interface makes available. The following segment of code, for example, is the part of OnBtnStart() in TranscodeClientDlg.cpp. The code will set up and start a transcode. The actual calls to the Filter Manager interface are highlighted.

```
long errorCode;

// Reset all transcoder components, Check Results.
errorCode = m_pIFilterMgr->Reset();
if (errorCode != 0)
{
    DoNotStartTranscode();
    return;
}
// Load registry settings. Check Results
errorCode = m_pIFilterMgr->Load();
if (errorCode != 0)
{
    DoNotStartTranscode();
    return;
}
// Cue all components. Check Results.
errorCode = m_pIFilterMgr->Cue();
if (errorCode != 0)
{
    DoNotStartTranscode();
    return;
}

// Start Encode/Transcode. Check Results
errorCode = m_pIFilterMgr->Start();
if (errorCode != 0)
{
    DoNotStartTranscode();
    return;
}

// set encoder state to started
EncoderState = esStarted;
```

Note that we always check the *result* return value to ensure that the method succeeded. COM-related errors raise exceptions, while the components themselves

return a long result which is typically set to 0, if successful, or a negative value on error. For a complete list of Filter Manager error codes, refer to Appendix B.

All of the Filter Manager calls listed in the above code have been discussed in previous sections.

Releasing the COM Libraries

Like all resources, the COM libraries must be released when the programmer is finished using them. For this purpose, a single method is provided by COM called **CoUnitialize**. The **CoUnitialize** method should be invoked as follows in the destructor of the client code:

```
…
CoUnitialize();
…
```

At this point, you may be wondering about releasing the interface pointers. The **_com_ptr_t** Smart Pointer class handles this for you. The **IArgusFMPtr** destructor is invoked when the Smart Pointer goes out of scope. In its destructor, it releases the interface pointer it encapsulates after calling the **Release** method on the interface.

Registering to Receive Filter Manager Events

When you're programming in C++, registering to receive events is not the easiest of tasks. However, ATL does provide a template class, IDispEventImpl, to assist C++ client applications in receiving events from the server.

Before registering to receive events, make certain that you have already called _Module.Init(NULL, AfxGetInstanceHandle()) to initialize the data members of the COM server module. This should be done when you initialize your application. Also remember to call _Module.Term() before exiting your application.

These are the steps you'll need to follow to register for events using the IDispEventImpl template:

1. Derive a class from the IDispEventImpl template. For example, in our sample application, we derive CFilterManagerEvents from IDispEventImpl:

```
class CFilterManagerEvents:
public IDispEventImpl<1, CFilterManagerEvents>
```

The first argument to IDispEventImpl is a unique identifier for the event source. Since Filter Manager is the only source from which our sample application receives events, its ID of 1 is, indeed, unique.

The second argument is a reference back to the deriving class.

2. In the class definition, insert a SINK_MAP that includes each of the events from Filter Manager that you'd like to receive. For example, in our sample application, we register to receive the error, log, finished, and pause events.

```
BEGIN_SINK_MAP(CFilterManagerEvents)
    SINK_ENTRY(1, 1, OnError)
    SINK_ENTRY(1, 2, OnLog)
    SINK_ENTRY(1, 3, OnFinished)
END_SINK_MAP()
```

The arguments for the SINK_ENTRY line are (SOURCE, DISPID, FUNC), where:

- SOURCE identifies the event source. Since Filter Manager was identified by a value of '1' in step 1, above, we use the value '1' here to indicate that we're receiving events from Filter Manager.
- DISPID identifies the dispatch ID within the event source of the event that we're receiving. In Filter Manager, the Error Event has a dispatch ID of 1, the Log Event has a dispatch ID of 2, the Finished Event has a dispid of 3, and the Pause Event has a dispid of 4.
- FUNC identifies the function or method that will handle the received event. This method will be defined within the class that we're currently deriving from IDispEventImpl.

3. Define and implement a method that calls AtlGetObjectSourceInterface() and DispEventAdvise(). First call AtlGetObjectSourceInterface() to retrieve *pUnk*, a pointer to the interface ID of the default source interface. Then call Disp-EventAdvise() to establish a connection with the event source represented by *pUnk*. This connection allows events fired from *pUnk* (or, in our case, from Filter Manager) to be routed to the handler functions specified in our event sink map.

In our C++ sample application, the EasyAdvise method is included in the CFilter-ManagerEvents class to perform the connections described in the paragraph above:

```
HRESULT EasyAdvise(IUnknown* pUnk)
{
    // Make sure the COM object corresponding to pUnk implements
    // IProvideClassInfo2 or IPersist*. Call this method to extract info
    // about source type library if you specified only 2 parameters to
    // IDispEventImpl
    HRESULT hr = AtlGetObjectSourceInterface
        pUnk, &m_libid, &m_iid, &m_wMajorVerNum,
        &m_wMinorVerNum);
    // connect the sink and source
```

```
    hr = DispEventAdvise(pUnk, &m_iid);
    return hr;
}
```

4. Within the CFilterManagerEvents class, define and implement a method that calls AtlGetObjectSourceInterface() and DispEventUnadvise(). Once again, AtlGetObjectSourceInterface() is called to retrieve *pUnk*, a pointer to the event source. DispEventUnadvise() breaks the connection with the event source represented by *pUnk*. Once the connection is broken, events will no longer be routed to the handler functions.

In our C++ sample application, the EasyUnadvise method is included in the CFilterManagerEvents class:

```
HRESULT EasyUnadvise(IUnknown* pUnk)
{
    AtlGetObjectSourceInterface (pUnk,&m_libid, &m_iid,
        &m_wMajorVerNum, &m_wMinorVerNum);
    return DispEventUnadvise(pUnk, &m_iid);
}
```

5. Within the CFilterManagerEvents class, define and implement the functions that will handle the events that you've registered for. Within the body of each of these event handlers, insert code to respond in whatever way you decide to the event that you're receiving. For example, you may choose to write out to a log file any messages that you receive from a Log Event.

Within the class definition of our sample application, the event handlers are prototyped as follows:

```
STDMETHOD(OnError)(long code, BSTR error);
STDMETHOD(OnLog)(long code, BSTR error);
STDMETHOD(OnFinished)(long code, BSTR error);
```

Following is an example of the implementation of an event handler. Note that we call MessageBox() for demonstration purposes only. **For deliverable applications, you should not hold up a log event — or any other type of event — with a function requiring user input.**

```
/*****************************************************************************
 * OnLog()
 *
 * Event sink called by Filter Manager fires a log event.
 *****************************************************************************/
```

```
STDMETHODIMP CFilterManagerEvents::OnLog(long code, BSTR error)
{
    CString strMessage = _T("");
    strMessage.Format ( _T("%s"), ( LPCTSTR ) error);
    pView->MessageBox( strMessage );
    if( !pFile )
        return S_OK;
    // Send error message to log file.
    if( code < 0 )
        _ftprintf(  pFile, _T("Warning %ld: %s\n"), code, strMessage );
    else if (code !=99)
        _ftprintf( pFile, _T("%s\n"), strMessage);
    fflush( pFile );
    return S_OK;
}
```

When you implement the event interfaces, it is important to consider what Prism software threads or processes will be firing them. Most events generated by the CineView Pro Prism API will be called in the context of their own thread, which is engaged in near real-time processing with the decoder hardware. If event interface methods cause excessive delays, exceptions to be raised, or the calling thread to wait indefinitely on a synchronization object, undesirable behavior is sure to result. If these types of operations need to be performed in response to an event, create your own thread, return control to the calling thread, and proceed to do the processing in the context of your own thread.

For example, if receiving a Finished event should trigger your application to start the next transcode, make certain that your implementation of Finished event just toggles a flag or a semaphore, which prompts another thread (perhaps the main thread) to cue the next transcode. This will ensure that the threads invoking the event interfaces remain responsive to hardware events.

It is also important to realize that, by implementing an event interface, you have effectively made your client code a server for those events. This means you need to take into account the synchronization of data accessed by objects using the event interface.

Running the Sample Application

The TranscodeClient (CVProPrism) sample C++ application, portions of which have been discussed in the previous sections of this chapter, is intended as an example of the use of the Filter Manager interface, not as an end-user application. However, it is useful to compile and run the sample application to observe and understand the operation of the CineView Pro Prism transcoder.

Performing a Transcode

The sample application offers a simple, straightforward user interface that demonstrates transcoding an MPEG asset into the desired format. After entering the full pathname of the file that will be transcoded, press the Start button, which invokes the OnBtnStart() method listed earlier.

When the Start button is clicked, OnBtnStart() does the following:

- Calls Reset() to clear the transcoding properties.
- Reads the text edit field values, and other editable fields from the window, and tabs and stores them in the Windows Registry.
- Calls the FilterManager Reset() method to reset all of the transcoder components.
- Calls the FilterManager Load() method to load all settings from the Registry.
- Reports an error and fails if the Load() method returns unsuccessfully.
- Calls the Filter Manager Cue() method to set up all of the transcoder COM components for a transcode.
- Checks the Cue() return code, aborting on error.
- Calls application-specific methods to intelligently set the appropriate buttons to enabled or disabled. Note that the Cue method does not return until the transcoder is ready to begin transcoding — or until an error occurs.

Now that the transcoder has completed its “cue” process, the Filter Manager Start() is called to begin the transcoding process. Once the transcode has begun, it continues until the specified MPEG file has ended, or until the user clicks the Stop button.

The result of the transcode will be a duplicate of the original MPEG file, but in the desired format (Ligos MPEG-1, Real G2, or Windows Media format).

If an error occurs during the transcoding process, the sample application will receive an Error Event. In response, it calls the Filter Manager Stop() and Reset() methods.

Windows Registry Overview

As discussed in Chapter 2, most of the properties that must be defined before starting the transcode process are set in CineView Pro Prism-specific Windows Registry tables. Many of these properties can be set once when the transcode software is installed and can be left unchanged thereafter. Specific applications, however, may need to reset a subset of the transcoder properties before each transcode. For example, most applications will assign a new file path name with each transcode. Other applications may allow the user to change the bit rate or resolution with each transcode.

In order to adjust specific Windows Registry settings prior to a transcode, you may need to control the Registry programmatically. To allow programmatic access to the transcoder Registry tables, Vela has designed a C++ class (CRegistry), the source code of which is provided in the TranscodeClient sample application. TranscodeClient uses the CRegistry class to read from and to write to the transcoder Registry tables. We encourage you to use the CRegistry class to access and modify transcoder Registry tables. Example screen shots developed from the TranscodeClient program are shown at the end of this section.

CRegistry Methods

The CRegistry class provides the following five methods (or, in the case of numbers 3 and 4, types of methods).

1. **Constructor**: Creates an instance of the class and initializes its members.

2. **Open**: Opens the Registry. Returns TRUE if successful, FALSE otherwise.

3. **SetValue**: Writes a setting to the Registry. There are a number of SetValue methods defined in the CRegistry class, each of which handles a specific data type. The SetValue() method is usually called with two arguments: The name of the Registry key and the value to be saved to that Registry key. The method returns TRUE if it is successful.

4. **GetValue**: Reads a setting from the Registry. Again, there are a number of GetValue methods defined in the CRegistry class, each of which handles a specific data type. The GetValue method is usually called with three arguments: The name of the Registry key, a pointer to a variable to hold the value read, and a default value to be given to the variable if no Registry setting is available. The method returns TRUE if it is successful.

5. **Destructor**: Closes the Registry table.

Example: Loading a Transcoder Registry Table

As an example of using the CRegistry class to load settings, the following method loads some of the transcode settings from the Registry table. Again, this method is provided as an example of the calls required to read a table from the Registry. In a production-quality application, you should check the return value of each of the included function calls to make certain that the value is equal to TRUE.

```
void CTransForm::InitializeTranscodeSettings()
    {
        CRegistry Settings;
        CRegistry DualEnc;

        // Initializing Transcode Settings according to registry
        if( Settings.Open(HKEY_CURRENT_USER, ARGUS_KEY ) == TRUE )
        {
                if( DualEnc.Open(Settings.hKey(), _T("DualEnc")) == TRUE )
                {
                        DualEnc.GetValue( _T("Filename"), &m_csFilename,
                        _T("D:\\MpegFiles\\test.mpg"));
                        DualEnc.GetValue( _T("HorizSize"), &m_lHorizRes,
                        176 );
                        DualEnc.GetValue( _T("VertSize"), &m_lVertRes,
                        120 );
                        DualEnc.Close();
                }
                Settings.Close();
        }
    }
```

Example: Storing Values in a Transcoder Registry Table

As an example of using the CRegistry class to store settings, the following function saves some of the video transcode settings in the "DualEnc" Registry table. Once again, in a production-quality application, you make certain that each call returns TRUE before continuing to the next. Note that the Settings and DualEnc Registry sections are closed by the destructor when the Settings and DualEnc objects go out of scope.

```
void CTransForm::SaveTranscodeSettings()
    {
        UpdateData();
```

```
        CRegistry Settings ;
        CRegistry DualEnc ;

        if( m_lHorizRes == 176 ) // QSIF
        {
                m_lVertRes = (vidMode == VIDEO_MODE_NTSC) ? 120 : 144;
        }
        else
        {
                m_lVertRes = (vidMode == VIDEO_MODE_NTSC) ? 240 : 288;
        }

        // Save Transcode Settings to registry
        if( Settings.Open(HKEY_CURRENT_USER, ARGUS_KEY ) == TRUE )
        {
                if( DualEnc.Open(Settings.hKey(), _T("DualEnc")) == TRUE )
                {
                        DualEnc.SetValue( _T("Filename"), m_csFilename );
                        DualEnc.SetValue( _T("HorizSize"), m_lHorizRes );
                        DualEnc.SetValue( _T("VertSize"), m_lVertRes );
                        DualEnc.SetValue( _T("InAudioSampleFreq"),
                        m_nInputAudioSampleRate);
                        DualEnc.SetValue( _T("ChromaFormat"),
                        m_radChroma422Ctrl.GetCheck() );
                        DualEnc.SetValue( _T("NTSC"),
                        m_radNTSCCtrl.GetCheck());

                        DualEnc.Close();
                }
                Settings.Close();
        }
}
```

For More Information on Registry Control

For more examples of the use of the CRegistry class, review the source code for the TranscodeClient application. In addition to defining CRegistry, this project has a .cpp file to manage each transcoder Windows Registry table. For information about each of the Registry settings, refer to Appendix A of this document.

CVProPrism Typical Screen Shots

When you double-click on CVProPrism.exe, tabbed windows will appear, as shown on this and the following pages. Registry changes can be made as desired. To restore the particular tabbed section to factory default settings, click the "Set Default" button.

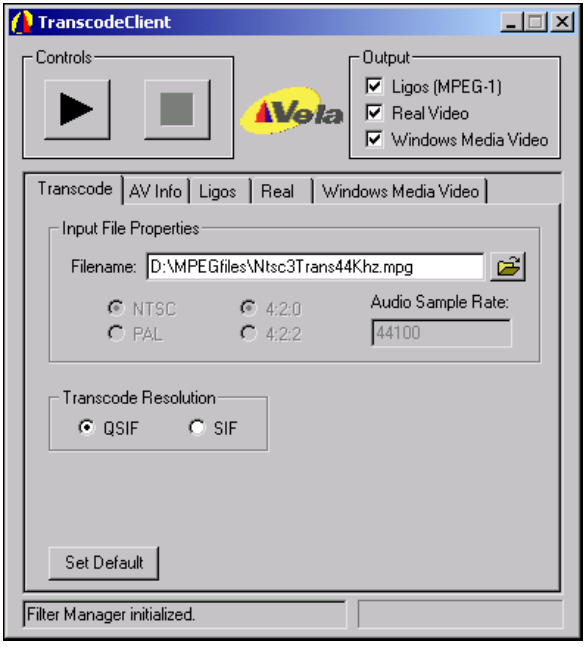

Figure 3-1. Registry Control Panel — Transcode

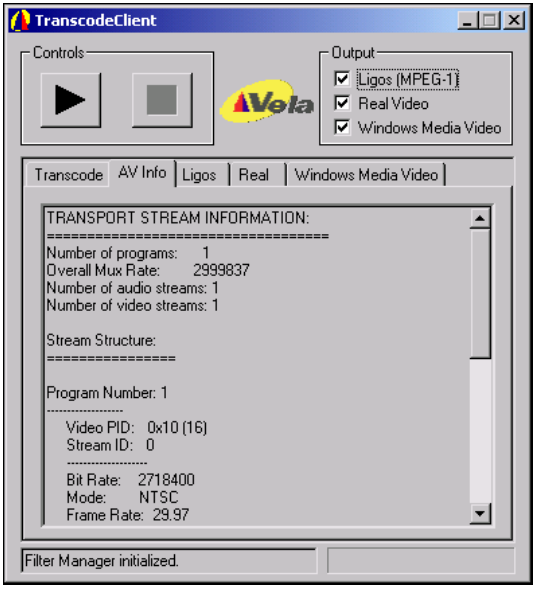

Figure 3-2. Registry Control Panel — A/V Info

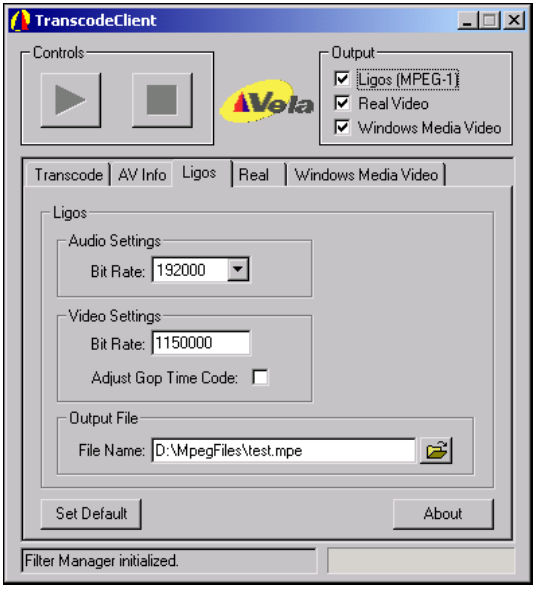

Figure 3-3. Registry Control Panel — Ligos

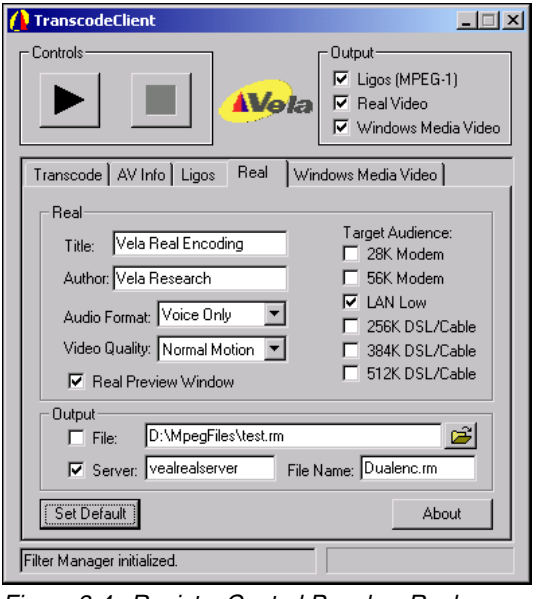

Figure 3-4. Registry Control Panel — Real

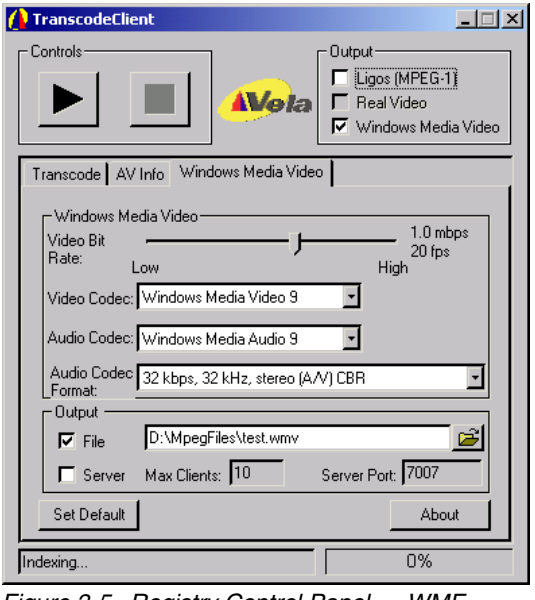

Figure 3-5. Registry Control Panel — WMF

Chapter 4

Distributing Components

Overview

Building an installation disk is very important. This is the first view a user will have of your software system. It is crucial that the installation procedure install your software easily and correctly. The last thing you want to hear from a user is that the software won't install.

For the Vela CineView Pro Prism version 2.6 we use Wise® version 8.1 to build our software installation package. The resulting installation disks are robust and easy to follow, and they give a nice presentation to your package. All of the necessary steps required by our components (driver and component registration, Windows Registry setup, etc.) are handled automatically by this software. If this product meets your needs, use it. If it does not, there are many alternative installation products.

The concept of this chapter is to give you some direction on creating your installation disks. It is very important that certain features of the component architecture be installed correctly in order for the Prism transcoder to function properly. You will, of course, be required to add the portions that you have created (any *.EXE files and required *.DLL files) to the install script. We list the files that are needed for your install script, and where they can be found. It is important that these files be placed into the same directory structure on the destination machine.

The following issues must be addressed within the installation procedure in order for the board(s) and components to function correctly:

- Driver installation/Registry settings
- Redistributable files
- Microcode directory structure
- Required COM components and registration

Driver Installation and Registry

The CineView Pro Prism uses the following driver:

- **SAA_7146.SYS**, the decoder device driver.

The device driver has associated Registry settings that provide Windows with information about the driver. The Registry settings are listed below in an .INI format. If you have the soft copy of the file, you may simply cut and paste each

section into an individual .INI. Using a utility such as Regini will help automate the Registry settings for you.

When creating your Registry installation, make sure to account for all spaces and punctuation listed in the Registry settings below. Incorrect formatting is the most common reason for custom installations to fail.

Decoder Real-Time Playback

Please refer to Chapter 3 of the CineView Pro/Pro LE Installation and User's Manual for a complete list of drivers and system DLLs that the playback software requires.

Microsoft Redistributable Code

The current installation requires two sets of Microsoft Redistributable Code:

- MFC Class Libraries:

  MFC42.DLL

  MFC42U.DLL

  MSVCRT.DLL

  WININET.DLL

- COM Registration:

  ATL.DLL

  COMCTL32.DLL

  COMCTL32.OCX

  OLEPRO32.DLL

  REGSVR32.EXE

File names may differ depending upon the version used. See the online help for Microsoft Developers for more information. This list can also be found on the Microsoft online help files for Distributing ActiveX Controls. In addition to the core registration files, the file ATL.DLL needs to be added to allow the COM objects to self-register.

For CineView Pro/Pro LE components, refer to the *CineView Pro/Pro LE Installation/User Manual and API Guide.*

Microcode Directory Structure

The CineView Pro decoder also has microcode requirements. CinProSerCom.dll uses the microcode files, installed during the CineView Pro installation process, in the C:\Microcode directory to initialize the CineView Pro hardware. For more information on these files, refer to the CineView Pro product documentation.

COM Components

The following COM objects, which comprise the API set, are distributed with the standard Vela CineView Pro Prism application. These COM components are registered automatically as part of the installation process.

CineView Pro Prism COM components located in:
C:\Program Files\Vela Research\CineView Pro\Prism:

- CINPROSERCOMU.DLL
- FILTERMANAGERU.DLL
- MULTIPLEXU.DLL
- REMOTESTOREU.DLL

CineView Pro Prism COM components located in:
C:\Program Files\Vela Research\Common

- MEMORYMANAGER.DLL
- MEMMGRSERVER.EXE
- VELA_PINS.DLL

CineView Pro Prism COM component located in:
C:\Program Files\Vela Research\Common\MPEG:

- LIGOSENCODEU.DLL

CineView Pro Prism COM components located in:
C:\Program Files\Vela Research\Common\WMF:

- ASFENCODEU.DLL
- ASFWRITERU.DLL
- wmfdist.exe (Executing during installation will copy and register necessary files to the user system.)

CineView Pro Prism COM components located in:
C:\Program Files\Vela Research\Common\G2:

- 14_43260.dll
- 28_83260.dll
- atrc3260.dll
- auth3260.dll
- basc3260.dll
- cook3260.dll
- dnet3260.dll
- edn t3260.dll
- enceng.lib
- encn3260.dll
- enlv3260.dll
- erv13260.dll
- erv23260.dll
- erv33260.dll
- espr3260.dll
- pnrs3260.dll
- realencode.dll
- rmme3260.dll
- rmto3260.dll
- rmtools.lib
- rn5a3260.dll
- rnco3260.dll
- rv103260.dll
- rv203260.dll
- rv303260.dll
- sdpp3260.dll
- sipr3260.dll
- tokr3260.dll

Transcoding Files

The following files are required to perform each of the three types of transcoding. Note that these third-party files require licensing from their manufacturers before redistribution.

HASP COM component located in:
C:\Program Files\Vela Research\Common\hasp

- hinstall.exe (Executing during installation will copy and register necessary files to the user system.)

Ligos COM components located in:
C:\WINNT\system32:

- gmdvsd.dll
- GMVFWCAP.dll
- gomotion.dll
- LsxMpgk7.dll
- LsxMpgp2.dll
- LsxMpgp3.dll
- LsxMpgp4.dll
- LsxMpxp2.dll
- LsxMpxp3.dll
- LsxMpxp4.dll

RealNetworks COM components located in:
C:\WINNT\system32:

- pncrt.dll
- pngu3266.dll
- rmbe3260.dll

Component Registration

In order for Prism software to run, all of the COM components listed above must be registered. (On the other hand, executable servers like MemMgrServer.exe and CineViewServer.exe register themselves at runtime and destroy themselves when they are no longer in use.) The Prism installation procedure registers all of the standard COM components. The SDK uses the same registered DLLs as the standard application.

If Vela CineView Pro Prism COM components are installed on a system without an automatic registration program like Wise, you can register them using the REGSVR32.EXE application provided with the Prism. To register a COM component, type

```
Regsvr32 /s <COM component filename>
```

Regsvr32 is a utility provided at no extra charge by Microsoft that you are free to redistribute.

Appendix A

Transcode Registry Settings

Overview

All of the Registry settings used to configure an individual transcode on the Vela CineView Pro Prism are found in one of the six Registry locations listed below. Refer to “System Configuration Settings,” page 10, for additional system-level Registry settings.*

Standard Vela Prism Registry Tables:

\HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\DualEnc

\HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\FilterMgr

\HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\LigosMpeg1

\HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\LigosMux

\HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\RealNetworks

\HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\WMF

If the Prism Registry locations listed above are not established prior to the first transcode, a call to the Filter Manager method Load() will create the Registry, providing default settings for each of the keys. These settings can be modified programmatically (refer to MSDN help) or manually using the **regedt32** or **regedit** command.

The Filter Manager component provides two special functions that load and save all of the Filter Manager settings as well as those of the tables listed above:

long Load() – Loads all of the Registry settings for the locations listed above. If the Registry location does not exist, the Load() call creates it, creates all of the Registry keys, and assigns them their default values. This method should be called prior to each transcode. Returns 0 if successful, or, on failure, returns an error code listed in Appendix B.

long Save() – In the appropriate Registry tables, saves all of the settings for the current transcode. Returns 0 if successful, or, on failure, returns an error code listed in Appendix B.

---

*Where the data type of the key is listed on the charts in Appendix A, you should assume that all integral types should be stored in the Registry as keys of type REG_DWORD. All CString values should be stored as keys of type REG_SZ.

Refer to "Changing Individual Registry Settings," page 12, for an explanation and examples of manipulating transcoder Registry settings programmatically with the CVProPrism sample application included with this application.

Setting the Registry Tables

Descriptions of the keys found in each of these Registry locations are listed in the following tables. CVProPrism screen shots can be found at the end of Chapter 3.

Second Audio Channel in Secondary Stream (CineViewPro XL)

If you are using a CineViewPro XL with Vela's Argus Spectrum encoder, it is possible to send the second audio channel to the secondary stream's audio source rather than using the default primary audio channel. Except for the Pro XL, no other member of the CineView Pro decoder family can be used for second channel audio while multi-streaming. There are two registry settings that must be set to enable this feature.

First a registry value must be set in the CineView Pro Key that will enable the generation of the second audio pin file. This registry setting is different from other Spectrum settings because you must modify the registry key HKEY_CURRENT_USER\Software\Vela Research\CineViewPro 0\Settings instead of the standard "Broadcast Argus" key.

CineView Pro XL Registry Table				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Second Audio Channel Capture	SecondAudioCaptureEnabled	Int / REG_DWORD	Disabled=0 Enabled=1	Default is Disabled

Table A-1. CineView Pro XL Registry Table

Secondly, a registry value in the DualEncode key must be set in order to tell the secondary codecs to use the second audio pin. The full key name is: HKEY_CURRENT_USER\Software\Vela Research\Broadcast Argus\DualEnc; the value to set is SecondAudioCaptureEnabled. See Table A-2, "DualEnc Registry Table."

In order to enable the second audio capture, the CineView Pro "SecondAudioCaptureEnabled" Registry key must be enabled prior to starting the application. The registry property in the Argus key controls which audio channel will be used in the secondary stream on a per clip basis.

The *DualEnc* Registry Table

When performing a transcode, you'll need to set up the *DualEnc* Registry table before calling the FilterManager Load() and FilterManager Cue() methods. The *DualEnc* Registry table holds three keys that turn on or off each of the transcoding components, two keys that define the horizontal and vertical resolutions of the output streams (Ligos, Real, and/or Windows Media formats), and one key to set the file to be transcoded.

DualEnc Registry Table				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Ligos Flag	LigosEnabled	Int	TRUE (1) if a Ligos stream is to be created, FALSE (0) if not. Default: 0	The setting of this key is ignored if either the Ligos software module or the HASP hardware device is not installed on the computer.
Real Flag	RealEnabled	Int	TRUE (1) if a RealNetworks stream is to be created, FALSE (0) if not. Default: 0	The setting of this key is ignored if either the Real software module or the HASP hardware device is not installed on the computer.
Windows Media Format Flag	AsfEnabled	Int	TRUE (1) if a Windows Media-formatted stream is to be created, FALSE (0) if not. Default: 0	The setting of this key is ignored if either the Windows Media software module or the HASP hardware device is not installed on the computer.
Horizontal Resolution of Stream(s)	HorizSize	Long	352 (SIF) or 176 (QSIF). Default: 352	Note that more CPU power is required to support SIF than to support QSIF.

Table A-2. DualEnc Registry Table

DualEnc Registry Table (Continued)				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Vertical Resolution of Stream(s)	VertSize	Long	NTSC: 240 (SIF) or 120 (QSIF) Default: 240 PAL: 288 (SIF) or 144 (QSIF). Default: 288	Note that more CPU power is required to support SIF than to support QSIF.
Input Filename	Filename	CString	The full path name of the file to be transcoded.	
Input File: Vertical Resolution	InputVertSize	Long	480 (NTSC Full) 512 (NTSC VBI) 576 (PAL Full) 608 (PAL VBI)	
Second Audio Channel Capture	SecondAudioCaptureEnabled	Int / REG_DWORD	Disabled=0 Enabled=1	Default is Disabled

Table A-2. DualEnc Registry Table (Continued)

If you assign a value of 1 to the Ligos Flag, the Real Flag, or the Windows Media Format Flag in the *DualEnc* Registry table, you'll also need to configure the Registry table pertaining to the activated stream type. The next three sections of this document describe in detail the key settings for Ligos, Real Networks, and Windows Media Format transcoding.

The *Ligos Mux* Registry Table

The Filter Manager automatically sets all of the Ligos Mux table keys except AdjustGOPTimeCode, which the application developer may set to either 0 or 1, and GopTcStart, which is defined below.

Mux Registry Table				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Adjust GOP Time Code Flag	AdjustGopTime-Code	Unsigned char	0 = Off 1 = On	A setting of 1 tells the transcoder to stamp the GOP time codes to correspond to the tape deck times, starting with the mark-in value.
Starting GOP Time Code	GopTcStart	Unsigned long	Default: 0	If the adjust GOP time code flag is set to 1, this key identifies the starting time code. See note (1).
(1) The starting time code is an unsigned long of the format t:hh:mm:ss:ff, where the high-order digit "t" represents the time code type (0 = PAL, 1 = NTSC, 2 = drop-frame NTSC); the "hh" digits represent the hours field of the time code, the "mm" digits represent the minutes field, the "ss" digits represent the seconds field, and the "ff" digits represent the frames field. For example, a drop-frame starting time code of 01:32:43:14 would be represented as 201324314.				

Table A-3. Ligos Mux Registry Table

Setting the Registry for Ligos® Transcoding

If you want your output stream to be in MPEG-1 format, select the *LigosEnabled* option in the *DualEnc* Registry table. When this feature is turned on, your transcoding session will result in an MPEG-1 stream. Encoded by the Ligos GoMotion MPEG-1 software encoder and muxed by Vela's proprietary Mux software, the MPEG-1 stream starts on precisely the same frame as the MPEG-2 stream, matching its content frame-by-frame to the end of the transcode. (The Ligos stream will end one frame before the primary stream.) It can be transcoded in SIF (352 x 240/288) or QSIF (176 x 120/144) resolutions, as defined in the *DualEnc* Registry table, with the QSIF requiring less CPU usage than the SIF.

Use the Registry keys in the *LigosMpeg1* table (Table A-4) to set most of the properties for the Ligos MPEG-1 output stream. Note that the audio sample frequency for the MPEG-1 stream will be set automatically to that of the MPEG-2 stream. However, note that the Ligos encoder does not support a sample rate of 32,000.

LigosMpeg1 Registry Table				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Ligos File Name	Filename	CString	The full path name of the file that will hold the MPEG-1 stream.	
Audio Bit Rate	AudioBitrate	Long	64,000 80,000 96,000 112,000 128,000 160,000 192,000 224,000 256,000 320,000 384,000 Default: 192,000	
Video Bit Rate	TargetBitrate	Long	512,000 to 3,000,000 Default: 1,150,000	

Table A-4. LigosMpeg1 Registry Table

Setting the Registry for RealNetworks® Transcoding

If you want at least one of your output streams to be in RealNetworks format, select the *RealEnabled* transcoding option in the *DualEnc* Registry table. When this feature is turned on, your transcoding session will result in a RealNetworks-formatted stream. It can be transcoded in SIF (352 x 240/288) or QSIF (176 x 120/144) resolutions, as defined in the *DualEnc* Registry table, with QSIF requiring less CPU usage than SIF.

One useful feature of producing an output stream in Real format is that the stream can be displayed during the transcode as a confidence monitor on your screen. To enable this feature, set the *PreviewEnabled* flag, described in the table below, to 1.

RealNetworks Registry Table				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Real-time VGA Playback Flag	PreviewEnabled	Int	TRUE (1) to turn on the Real display during the transcode. FALSE (0) to turn off the display.	Turning on the display of RealNetworks video during the transcode does require additional CPU usage. You may need to adjust other parameters to accommodate.
Store File Flag	FileEnabled	Int	TRUE (1) if the Real stream is to be stored on the local drive. FALSE (0) if no local file is to be created.	If set to 1, define the path name of the local file in Filename.
Real File Name	Filename	CString		The full path name of the file that will hold the RealNetworks stream, if FileEnabled is set to 1. Default: D:\\Mpegfiles\\DualEnc.rm.

Table A-5. RealNetworks Registry Table

RealNetworks Registry Table (Continued)				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Stream Data Flag	ServerEnabled	Int	TRUE (1) if the RealNetworks file is to be streamed to a remote server.	If set to 1, define the name of the server in ServerName and define the path name of the remote file in ServerFilename.
Name of RealNetworks server	ServerName	CString	The name of the computer that will receive and store the RealNetworks stream, if ServerEnabled is set to 1.	This machine must be configured as a RealNetworks server.
Remote File Path	ServerFilename	CString	Name of the file to be streamed to the RealServer, if ServerEnabled is set to 1. No path is needed. The location of the file on the server is determined by the setup of the RealServer.	
Audio Content Type	AudContent	Int	Voice (0) Voice Background (1) Music (2) Music Stereo (3)	
Video Quality Setting	VidQuality	Int	Normal (0) Smooth Motion (1) Sharp Video (2) Slide Show (3)	Select the video quality that you require. Improved quality may required more CPU usage.

Table A-5. RealNetworks Registry Table (Continued)

RealNetworks Registry Table (Continued)				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
	Connect28K	Bool	On (1) or Off (0)	The audience determines the quality and frame rate of your audio and video. Improved quality may required more CPU usage. Selecting two or more at once automatically turns on SureStreaming, which will also increase CPU usage.
	Connect56K	Bool	On (1) or Off (0)	
	ConnectLAN	Bool	On (1) or Off (0)	
	Connect256K	Bool	On (1) or Off (0)	
	Connect384K	Bool	On (1) or Off (0)	
	Connect512K	Bool	On (1) or Off (0)	
Stream Title	Title	CString	The title of the transcoded stream.	
Author	Author	CString	Author of the transcoded material.	

Table A-5. RealNetworks Registry Table (Continued)

Setting the Registry for Windows Media™ Transcoding

If you want at least one of your output streams to be a Windows Media-formatted file (previously known as ASF), select the *AsfEnabled* transcoding option in the *DualEnc* Registry table. When this feature is turned on, your transcoding session will result in a stream in WMV format. It can be transcoded in SIF (352 x 240/288) or QSIF (176 x 120/144) resolutions, as defined in the *DualEnc* Registry table, with QSIF requiring less CPU usage than SIF. Other resolutions are valid but may not be valid for other output streams. Some other valid WMF resolutions include 200 x 200, 240 x 200 and 120 x 100.

WMF Registry Table				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
WMF File Name	Filename	Cstring / REG_SZ	The full path name of the file that will hold the Windows Media-formatted stream. Default: D:\Mpeg-files\test.wmv.	
Video Codec Type	VideoCodec	Int / REG_DWORD	ISO MPEG-4 ver. 1 = **0** Microsoft MPEG-4 ver. 3 = **1** Microsoft Windows Media™ Video ver. 7 = **2** Microsoft Windows Media Video ver. 8 = **3** Windows Media Screen ver. 9 = **4** Windows Media Video ver. 9 = **5**	The codec that provides the most quality at the lowest bandwidth is the Windows Media Video ver. 9 codec.
Video Bit Rate	VideoBitRate	long / REG_DWORD	28,000 to 3,000,000 bits per second.	

Table A-6. WMF Registry Table

WMF Registry Table (Continued)				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Frames per Second	FrameRate	Int / REG_DWORD	10 to 30	30 most closely approximates NTSC, but requires highest CPU usage. Should probably set to 15 if running all three output stream types. The frame rate is dependent on the video bit rate in such a way that if the frame rate is too high for the specified bit rate frames will start to drop in order to maintain the video bit rate.
Audio Codec	AudioCodec	Int / REG_DWORD	Microsoft Windows Media Audio ver. 9 = **0** Spiro Labs ACELP (voice only) = **1** Microsoft Windows Media Audio Voice ver. 9= **2**	Use Microsoft Windows Media ver. 9 Voice codec for low-bandwidth voice-only streams.
Audio Codec Format	AudioCodec-Format	Cstring / REG_SZ		String represents the audio codec format string to use for the corresponding audio codec. See Table A-7 for possible values.

Table A-6. WMF Registry Table (Continued)

WMF Registry Table (Continued)				
Property	**Registry Key**	**Data Type**	**Value Set**	**Comments**
Server Enabled	ServerEnabled	long / REG_DWORD	Enabled=1 Disabled=0	Enables/Disables server functionality. Enabling allows data to be streamed to Windows Media Player or a Windows Media Server. To connect to data stream from Windows Media Player, select File > Open URL and type "HTTP://HostMachineName:1234," where HostMachineName is the Network ID of the Encoder and 1234 is the server's port number.
Server Port	ServerPort	long / REG_DWORD	Any valid network port number	Clients must specify a machine name and this port to connect.
Maximum Clients	MaxClients	long / REG_DWORD	0 to 99	Sets maximum number of clients allowed to connect directly to this machine.

Table A-6. WMF Registry Table (Continued)

Table of Audio Codec format String Values The following settings are valid for codec type specified
Valid settings for Microsoft Windows Media™ Audio version 9.0
“8 kbps, 8 kHz, mono (A/V) CBR”
“6 kbps, 8 kHz, mono (A/V) CBR”
“5 kbps, 8 kHz, mono (A/V) CBR”
“12 kbps, 8 kHz, stereo (A/V) CBR”
“10 kbps, 11 kHz, mono (A/V) CBR”
“8 kbps, 11 kHz, mono (A/V) CBR”
“20 kbps, 22 kHz, mono (A/V) CBR”
“16 kbps, 22 kHz, mono (A/V) CBR”
“32 kbps, 22 kHz, stereo (A/V) CBR”
“22 kbps, 22 kHz, stereo (A/V) CBR”
“20 kbps, 22 kHz, stereo (A/V) CBR”
“20 kbps, 32 kHz, mono (A/V) CBR”
“48 kbps, 32 kHz, stereo (A/V) CBR”
“40 kbps, 32 kHz, stereo (A/V) CBR”
“32 kbps, 32 kHz, stereo (A/V) CBR”
“48 kbps, 44 kHz, mono (A/V) CBR”
“32 kbps, 44 kHz, mono (A/V) CBR”
“20 kbps, 44 kHz, mono (A/V) CBR”
“320 kbps, 44 kHz, stereo (A/V) CBR”
“256 kbps, 44 kHz, stereo (A/V) CBR”
“192 kbps, 44 kHz, stereo (A/V) CBR”
“160 kbps, 44 kHz, stereo (A/V) CBR”
“128 kbps, 44 kHz, stereo (A/V) CBR”

Table A-7. Table of Audio Codec Format Strings

"96 kbps, 44 kHz, stereo (A/V) CBR"
"80 kbps, 44 kHz, stereo (A/V) CBR"
"64 kbps, 44 kHz, stereo (A/V) CBR"
"48 kbps, 44 kHz, stereo (A/V) CBR"
"32 kbps, 44 kHz, stereo (A/V) CBR"
"192 kbps, 48 kHz, stereo (A/V) CBR"
"160 kbps, 48 kHz, stereo (A/V) CBR"
"128 kbps, 48 kHz, stereo (A/V) CBR"
"96 kbps, 48 kHz, stereo (A/V) CBR"
"64 kbps, 48 kHz, stereo (A/V) CBR"
Valid settings for Microsoft Windows Media™ Audio Voice version 9.0
"4 kbps, 8 kHz, mono"
"5 kbps, 8 kHz, mono"
"8 kbps, 8 kHz, mono"
"10 kbps, 11 kHz, mono"
"12 kbps, 16 kHz, mono"
"16 kbps, 16 kHz, mono"
"20 kbps, 22 kHz, mono"
Valid settings for Spiro Labs ACELP Codec
"5 Kbits/s, 8000Hz, Mono"
"6.5 Kbits/s, 8000Hz, Mono"
"8.5 Kbits/s, 8000Hz, Mono"
"16 Kbits/s, 16000Hz, Mono"

Table A-7. Table of Audio Codec Format Strings (Continued)

Appendix B

Filter Manager Error/Status Codes

The following return codes may be returned by calls to Filter Manager methods or by Filter Manager Error Events. Please note that this listing was developed for the Argus encoder and includes some error codes that do not apply to the Vela CineView Pro Prism transcoder.

NOTE: If you abort a transcode or shut down your application without cleanly ending a transcode, you must make certain that CVProServer and MemMgrServer have both been terminated before you restart the application. You can terminate these services using Task Manager.

Filter Manager Error/Status Codes		
Error Code	**Meaning**	**Comments**
11	Encode has been successfully resumed after a pause.	Not an error.
0	Operation completed successfully. Status OK.	Not an error.
-9	Unable to eject tape using Sony command.	
-10	Video component failed to start, stop, pause, resume, or reset.	Summary error message. Check log for more descriptive messages regarding status of video encoder.
-12	Mux component failed to start, stop, pause, resume, or reset.	Summary error message. Check log for more descriptive messages regarding status of mux.
-13	Main storage component failed to stop, pause, or resume.	Summary error message. Check log for more descriptive messages regarding status of storage component.
-14	CineView Pro component failed on a stop.	May be locked up. Check log for other error messages.
-15	VTR component failed to cue, stop, pause, or write adjustment to Registry.	Summary message. Check log for other VTR-related messages.

Table B-1. Filter Manager Error/Status Codes

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-16	Storage component for elementary audio or video stream failed to stop or resume.	Summary message. Should occur only during elementary stream encode.
-17	Disk space error.	Application determined that there is insufficient disk space for the muxed file or for the elementary video file.
-18	VSP component failed during a reset.	Summary message. Check for more specific information from VSP component.
-25	Exception thrown during reset, cue, or start.	
-26	Failure to create one of Filter Manager mutexes.	Probably a system error. Check number of open handles.
-27	Filter Manager failed when trying to initialize COM libraries.	System problem? COM DLLs not installed?
-28	Real or WMF component failed to start.	Summary message. Check for more specific messages from Real or WMF.
-29	CineViewPro component failed to reset, cue, or start.	Summary message. Check for more specific messages from decoder component.
-30	No mux filename was supplied, so Filter Manager failed to create a codec file name.	Check file name field to make sure that it contains a legitimate file name.
-31	Exception thrown loading parameters from the Registry.	Summary message. Check Registry to see if table is there.
-32	No longer used.	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-33, -34	Error resetting or starting audio storage component.	Summary message. Check log file for more specific messages. Last encode may not have shut down correctly? Exit application and try again.
-35	Video storage, Ligos storage, or plug-in component failed to start.	Summary error. Check for more specific error messages in log.
-36	Main storage component failed to start or reset.	Summary message. Check for more specific messages in error log.
-39	Unable to set outpoint on tape deck.	Check COM port connection (including converter). Make sure tape deck is on remote. Make sure application is looking at the correct COM port.
-41	Invalid VTR shuttle speed requested through Sony protocol.	
-42	Tape deck failed to receive shuttle command.	Check COM port connection (including converter). Make sure tape deck is on remote. Make sure application is looking at the correct COM port.
-44	Invalid pre-roll value. Pre-roll should be a number between 0 and 60 (seconds). AND mark-in value must be greater than pre-roll.	Check validity of mark-in and preroll.
-45	When you adjust the mark-in time code by adding or subtracting the pre-roll adjustment, the result is invalid (<0).	Check mark-in and adjustment time codes. Adjustment must be less than mark-in.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-47	Unable to read preroll from tape deck.	Check COM port connections, Check to see that VTR is in remote mode, etc.
-48	Unable to read mark-in from tape deck.	Check COM port connections, Check to see that VTR is in remote mode, etc.
-49	Unable to recognize microcode type in looking up VTR adjustment in Registry.	
-50	Error initializing pin that connects IBMVideo to Mux component.	Check swap space on C:\ drive. Make sure mux was shut down properly last time (that CVProServer and MemMgrServer were not left running).
-51	Attempted to initialize a video component that was already initialized.	Try resetting or quitting application. Make sure no other instances of the encoder are running simultaneously. Be sure to terminate CVProServer and MemMgrServer before restarting.
-52	Attempted to cue a video component that was already cued.	Try resetting or quitting application. Make sure no other instances of the encoder are running simultaneously. Be sure to terminate CVProServer and MemMgrServer before restarting.
-53	Attempted to cue a video component that was already playing.	Terminate last encode (or wait for it to finish) before starting next encode.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-54	Attempted to Pause the video component while it was already paused.	Application may have lost track of its state. If error persists, you may need to exit application and restart.
-55	Attempted to Start the video component before cueing it.	Application appears to have lost track of its state. If error persists, you may need to exit application and restart.
-56	Attempted to Stop the video component, although it was not currently playing.	Application appears to have lost track of its state. If error persists, you may need to exit application and restart.
-57	Attempted to Resume the video component, although it was not currently paused.	Application appears to have lost track of its state. If error persists, you may need to exit application and restart.
-60	An exception was thrown from within the video process thread.	Check video settings (PAL/ NTSC, Digital/ Composite). Check video encoder hardware. Run video encoder diagnostics.
-61	Driver command to start video returned unsuccessfully OR exception was thrown by video Start method.	Check video settings (PAL/ NTSC, Digital/ Composite). Check video encoder hardware. Run video encoder diagnostics.
-70	Audio chip timed out during a read or shut-down command.	This is either a problem with the audio encoder hardware or with the system.
-71	Attempting to reinitialize an already-initialized audio component.	Shut down application. Make sure CVProServer and MemMgrServer have been terminated. Restart.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-73	Attempted to cue or Start an audio component that is currently encoding.	The application may have hung up during the previous encode. Try calling Reset. If that fails, restart the application.
-74	Attempted to Pause the audio component when it was already paused.	State problem? May need to restart the application.
-75	Attempted to Start the audio component before cueing it.	State problem? May need to restart the application.
-76	Attempted to Stop or Pause the audio component although it is not playing.	State problem? May need to restart the application.
-77	Attempted to Resume the audio component, although it is not paused.	State problem? May need to restart the application.
-78	Encode failed because audio pin over- flowed, or application was not successful in creating, initializing, or resetting the pin from the audio component to the mux.	If error occurred when application was coming up, check the C drive to be sure it has adequate swap space. If error occurred during an encode, the audio pin backed up-this is usually a secondary error. The audio backup is caused because some other component failed or "hung up" the encoder. Shut down application, terminating CVProServer and MemMgrServer if needed.
-79	A stop was issued to the audio component, but it won't stop in a reasonable amount of time.	It may be hung up in a while-loop. May need to terminate application with task manager.
-80	Software failed trying to read the firmware revision OR exception was thrown during the audio component initialization process.	Check audio encoder hardware. Make sure board is installed properly.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-81	Audio process failed because of a driver initialization error, an error reading data from the audio board, or for some other non-pin-related error.	Check audio encoder hardware. Make sure board is installed properly.
-82	The start-audio driver command failed or an exception was thrown during the audio Start method.	Check audio encoder hardware. Make sure board is installed properly.
-83	The init_audio driver command failed OR an exception was thrown during the audio Cue method.	Check audio encoder hardware. Be sure board and software are installed properly.
-98	While checking Registry for adjustment value, VTR encountered an invalid microcode type designator.	Check to be sure that full set of current software was installed successfully.
-113	Exception thrown by FTP component while streaming data.	
-114	No FTP server name was provided, or an exception was thrown trying to connect to FTP server.	Check Registry to make certain that the FTP server name was provided, if you asked for a streaming output. Then check FTP connections and server setup.
-115	Error establishing internet session for FTP transfer.	
-116	Either remote file name was not filled in or there was an exception thrown while trying to open the remote (FTP) file.	
-117	Error opening local storage file.	Check path name, folder permissions. Make sure file not already open.
-118	Error writing to or closing local file.	Check disk fullness and disk status. If this is a high-bitrate encode, it may be that the disk can't handle the throughput.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-119	Error initializing the input pin of the storage component.	
-121	Error creating the storage component process thread.	
-122	Encoder is unable to communicate through the serial port with the VTR.	Check the cabling from the VTR to the converter, from the converter to the serial port (make sure converter is not in backwards). Check the COM port setting on the encoder application (COM1 or COM2) and make sure it matches the number of the serial port being used. Make sure that the VTR is turned on and that it is in remote mode.
-165	Error communicating with the VSP.	
-166	Error creating CVspApi class.	Mismatched software components?
-170	Attempt to call audio Get or Put method with a stream index other than 0 or 1, or attempting to set invalid audio bitrate, invalid audio input type, audio layer, or audio headroom with a Put() call.	Programming error. Check source code.
-175	Error creating playback COM object.	Ascertain that CinProSer-Com is registered.
-177	Error creating VTR COM object.	Ascertain that VtrControl is registered.
-179	Error creating Audio COM object.	Ascertain that IBMAudio is registered.
-181	Error creating Video COM object.	Ascertain that IBMVideo is registered.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-183	Error creating audio elementary storage object.	Ascertain that RemoteStore is registered.
-185	Error creating video elementary storage object.	Ascertain that RemoteStore is registered.
-189	Error creating mux storage object.	Ascertain that RemoteStore is registered.
-194	Error initializing mux or Ligos mux object.	Ascertain that Multiplexor is registered.
-201	Error creating VSP object.	Ascertain that IBMVSP is registered.
-219	Error creating second audio object.	Ascertain that IBMAudio is registered.
-227	Error creating Ligos, Real, or WMF object.	Ascertain that LigosEncode, RealEncode, AsfEncode component is registered.
-230	During cue, mux component received an invalid stream type (0=system, 1=program, 2=transport, 3=elementary).	Check Registry setting of mux stream type.
-233	Error creating or initializing plug-in component's input pin.	-233 through -248 are all plug-in errors. Since they are working with the source code, developers should be able to track these error codes themselves.
-234	Attempted to initialize plug-in component when it was already initialized.	
-235	Error starting suspended plug-in process thread.	
-236	Attempted to start plug-in when it is already encoding.	
-237	Attempted to start plug-in without cueing it.	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-238	Attempted to stop the plug-in component when it was not playing.	
-239	Attempted to pause the plug-in component when it was already paused.	
-240	Attempted to resume the plug-in component when it was not paused.	
-241	Plug-in component failed while writing to file.	
-242	Plug-in failed to open file or allocate resources.	
-243	Attempted to cue plug-in when it was already cued.	
-245	Error creating one of the plug-in objects.	
-247	The Initialize() method failed for one of the plug-ins.	
-248	Summary error code. Filter Manager failed in Cueing, Starting, Stopping, Pausing, Resuming, or Resetting a plug-in.	
-250	A pin underflowed. The error message will indicate which pin.	This message indicates that one of the components is starved for data (it's not being delivered fast enough).
-251	A pin overflowed. The error message will indicate which pin.	Usually this is an indication that the system is not able to handle the volume of work that it is being asked to accomplish. Check task manager during encode to see where the bottleneck might be.
-252	Unable to find a matching reader/writer for a specified pin.	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-253	An attempt was made to read or to write too large a block of data to/from a pin. (Block was larger than pin size).	The error message will indicate which pin. This is a programming error.
-254	An attempt to create a pin object failed. The error message will indicate which pin.	Ascertain that Vela_Pin is registered.
-255	Pin component failed when trying to create a mutex	System error. Check handle count using task manager.
-260	Undefined error occurred when attempting to create or use a pin.	
-331	RemoteStore component attempted to write very last block of data to the file, but failed. This error occurs only when the FilterMgr "Optimized MuxWrite" flag is set to 1.	We use a write procedure that requires that all write-blocks must be evenly divisible by the disk sector size. To get around this restriction on the last block of data, we close the file, reopen it in another mode, then write the last block. This error could be a timing error-make certain that no attempt was made to move or lock the file before the encode finished.
-332	Error closing the remote (FTP) file.	Make certain that the FTP process was not aborted before the encode finished.
-334	Failure creating, initializing, or using the decoder input pin (usually from the Mux).	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-335	The PlayFromPin call to the CVPro Server failed. The decoder failed to start realtime playback.	Make sure that the previous encode did not end with an unclosed CVProServer or MemMgrServer executable running. Make sure that the decoder board is installed properly and functioning properly. Check firmware / hardware revisions of the decoder board.
-337	Attempt to set up CVPro scalar for dual-encode failed.	See Error Code -335.
-338	Attempted to stop the decoder when it was not playing.	
-340	An invalid closed-caption-type was defined (read from Registry).	See notes in Appendix A on closed caption types.
-341	Unable to create or initialize CVProServer object.	Ascertain that CVProServer is registered and functional.
-342	CVPro failed on request to initialize.	Check functionality of decoder in general using standard CVPro client application.
-343	CVProServer failed to pause.	
-344	CVProServer failed to resume.	
-345	When the mux Start component was called, there was no active thread running to start.	State problem. May need to restart application, making certain that CVProServer and MemMgrServer are terminated.
-346	Filter Manager asked mux component to create an undefined stream type. (See -230).	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-347	Mux component failed to open the mux writer stream.	
-348	An GOP Size of 0 or less was passed to Mux.	Check the I-frame distance setting in the Video Registry.
-349	Mux component failed to create its mutexes.	System problem. Check number of open handles using task manager.
-350	Mux failed when trying to initialize the closed-caption “driver” class.	
-400	Attempted to cue Ligos component when it was already cued.	State problem. May need to restart application.
-404	Attempted to Start Ligos when it was already playing.	State problem. May need to restart application.
-407	The call to Stop Ligos failed.	Component may be “hung up.” Try resetting. If that fails, application may need to be restarted.
-408	An attempt was made to reset the Ligos component when it was not installed. OR a call to reset Ligos, Ligos Mux, or Ligos Store failed.	See if the Registry shows the Ligos component turned ON even though Ligos is not installed or licensed on this system.
-410	Error encountered when trying to stop the Ligos Store component.	
-411	Error encountered trying to abort the Ligos Mux component.	May be “hung up,” in which case you'll have to terminate the application, along with CVProServer, and MemMgrServer.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-414	Attempted to set Ligos sample rate to 32,000, which is not supported.	Must use 44,100 or 48,000 sample rate. Sample rate must match that of primary MPEG-2 file.
-415	Failure to create one of 3 Ligos mutexes.	Check open handles using Task Manager.
-416	Ligos failed to create the video outpin or one of the two audio outpins.	Make certain Vela_Pins is registered.
-417	Failure setting up the Ligos software.	Make certain that Ligos component (and Ligos libraries) are properly installed.
-418	Failure to open the Ligos encoder.	Make certain that Ligos component (and Ligos libraries) are properly installed.
-419, -420	Error marshalling data into Ligos thread (for callbacks).	
-422, -423	Error opening the Ligos or codec Registry.	Use RegEdit and CVPro-Prism to check the existence and state of the Ligos and codec tables (See Appendix A).
-425	Error creating main Ligos processing thread.	
-426	Error writing data to the Ligos video out-pin.	This is most likely a pin problem. It could be that the processor can't handle all of the requests at this bitrate WITH the dual-encode running.
-427	Ligos thread unable to access pointer to main-process data and functions.	Timing error? Will probably need to restart application.
-428	Error initializing one of the three Ligos out-pins.	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-429	Invalid Ligos audio bitrate set in Registry. Must be 64000, 96000, 112000, 128000, 160000, 192000, 224000, 256000, 320000, or 384000.	Check Ligos Registry to ascertain that bitrate is one of those listed.
-431	Failed to create the in-pin from the CVPro to the Ligos component.	Ascertain that both CinProSerCom and LigosEncode are registered, as well as CVProServer. Make certain that previous encode closed down cleanly.
-432	Failed to initialize in-pin listed above.	
-433	Ligos failed to compress video frame.	May have received bad data from CVPro. Check content of tape and quality of MPEG-2 file, if possible.
-434	Ligos failed to compress audio frame.	May have received bad data from CVPro. Check content of tape and quality of MPEG-2 file, if possible.
-435	Ligos was given an invalid target bitrate.	Check LigosMpeg1 Registry for value of target bitrate. Must be between 500,000 and 3,000,000.
-436	Ligos was given an invalid horizontal resolution.	Check “DualEnc” Registry to make certain that horizontal resolution is either 176 or 352.
-437	Ligos was given an invalid vertical resolution.	Check “DualEnc” Registry to make certain that vertical resolution is either 240, 288, 120, or 144.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-438	A filename was not supplied in the Ligos Registry.	Check the LigosMpeg1 Registry to be sure that a file name was supplied. (For EDL Editor, this name will be generated based on the main MPEG file name.).
-440	Audio storage component (during elementary encode) failed to pause.	
-441	During elementary encode, video component failed to pause.	
-443	If a system, program, or transport stream is selected as the mux type in the mux Registry, but the mux-file-enabled flag is not set in the "FilterMgr" Registry table, this error flag is set. Also, if an elementary stream is selected as the mux type in the mux Registry, but the video-file-enabled flag is not set in the "FilterMgr" Registry, this flag is set.	Check stream type in Mux Registry, and compare it to the file type enabled in the FilterMgr Registry. See Appendix A.
-444	Invalid mux file path name.	Make sure that the pathname specified for the mux file is present and writable.
-445	Invalid video file path name.	For an elementary-stream encode, make sure that the pathname specified for the video file is present and writable.
-446	Invalid audio file path name.	For an elementary stream encode, make sure that the pathname specified for the audio file is present and writable.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-448	Argus Registry failure.	Unable to open the HKEY_CURRENT_USER path Software\Vela Research\Broadcast Argus. Check the Registry.
-449	Unable to open the “IBM Video” Registry table.	
-450	Unable to open the “IBM Audio” Registry table.	
-451	Unable to open the “Mux” Registry table.	
-452	Unable to open the “DualEnc” Registry table	
-455	Unable to open the “VTR” Registry table.	
-456	Unable to open the “RemoteStore” Registry table.	
-457	Unable to open the “FilterMgr” Registry table.	
-458	Invalid video bitrate supplied.	Check IBM Video Registry table (see Appendix A). Must be between 512,000 and 50,000,000. May not exceed 3,500,000 for SIF. May not exceed 15,000,000 for 4:2:0 chroma.
-459	Invalid horizontal resolution supplied for main encode. Must be 352, 480, 544, 704, or 720. (352 is the only valid value for SIF).	Check IBM Video Registry table (see Appendix A).
-460	Invalid vertical resolution supplied for main encode. Must be 120, 240, 480, 512 for NTSC or 144, 288, 576, 608 for PAL. (For SIF, must be 240 or 288).	Check IBM Video Registry table (see Appendix A).
-461	Invalid video mode supplied for main encode. Must be SIF (0) or AFF (1)	Check IBM Video Registry table (see Appendix A).

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-462	Invalid video format. Must be NTSC (0) or PAL (1).	Check IBM Video Registry table (see Appendix A).
-463	Inverse telecine is not supported.	
-464	Invalid input type supplied. Must be 1 for digital or any other value between 0 and 8 for composite.	Check IBM Video Registry table (see Appendix A).
-465	Invalid I-frame distance supplied in IBM Video Registry. Note that the I-Frame distance must agree with the RefFrameDistance and the Closed GOP flag.	Check IBM Video Registry table (see Appendix A) as well as section that immediately follows, also in Appendix A, explaining relationship between Closed GOP, I-frame distance, and ref-frame-distance.
-466	Invalid RefFrameDistance in IBM Video Registry table.	Check IBM Video Registry table (see Appendix A).
-467	Invalid ClosedGOP setting in IBM Video Registry table.	Check IBM Video Registry table (see Appendix A).
-468	Invalid chroma setting in IBM Video Registry table.	Check IBM Video Registry table (see Appendix A).
-469	Embedded metadata is not supported.	
-470	Invalid non-linear quantization setting. Must be 0 or 1.	Check IBM Video Registry table (see Appendix A).
-471	Invalid Concealment Vector setting. Must be 0 or 1. MUST set to 0 for low-bitrate SIF, or corrupted macroblocking will occur.	Check IBM Video Registry table (see Appendix A).
-472	Invalid DC Precision setting. Must be 8, 9, 10, or 11, with the value of 8 reserved exclusively for SIF encodes. Filter Manager will automatically encode SIF with a DCPrecision of 8, regardless of setting in Registry.	Check IBM Video Registry table (see Appendix A).

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-473	Invalid Intra-table flag.	Check IBM Video Registry table (see Appendix A).
-474	Invalid aspect ratio. Must be square(1), 4x3 (2), 16x9 (3) or 2.21 x 1 (4).	Check IBM Video Registry table (see Appendix A).
-475	Invalid audio bitrate supplied for main encode. Must be 32000, 48000, 56000, 64000, 80000, 96000, 112000, 128000, 160000, 192000, 224000, 256000, 320000, or 384000.	Check IBM Audio Registry table (see Appendix A).
-476	Invalid sample rate supplied for main encode. Must be 32000, 441000, or 48000. If Ligos encode is turned on, cannot be 32000.	Check IBM Audio Registry table (see Appendix A).
-477	Invalid audio mode supplied for main encode. Must be 0-Stereo, 1-Joint, 2-Dual, 3-Single.	Check IBM Audio Registry table (see Appendix A).
-478	Invalid audio input supplied for main encode. Must be analog, digital, or inactive.	Check IBM Audio Registry table (see Appendix A).
-479	Invalid audio layer for main encode. We support only layer 2.	Check IBM Audio Registry table (see Appendix A).
-480	Invalid error protect flag supplied for main encode. Must be 0 or 1.	Check IBM Audio Registry table (see Appendix A).
-481	Invalid copyright flag supplied for main encode. Must be 0 or 1.	Check IBM Audio Registry table (see Appendix A).
-482	Invalid “original” flag supplied for main encode. Must be 0 or 1.	Check IBM Audio Registry table (see Appendix A).
-483	Invalid slave-mode setting for main encode. Must be set to 1 if both audio AND video streams are enabled.	Check IBM Audio Registry table (see Appendix A).
-484	Invalid audio headroom setting for main encode. Must be 18 or 20.	Check IBM Audio Registry table (see Appendix A).

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-485	Invalid audio stream id for main encode. Must be a value of 0 to 31. May not duplicate stream id of other AUDIO streams in this encoded file.	Check Mux Registry table (see Appendix A).
-486	Invalid video stream id for main encode. Must be a value between 0 and 15.	Check Mux Registry table (see Appendix A).
-487	Invalid mux stream type for main encode. Must be system (0), program (1), transport (2) or elementary (3).	Check Mux Registry table (see Appendix A).
-488	Invalid language setting for one of audio streams in main encode.	Check Mux Registry table (see Appendix A).
-489	Invalid audio PID for one of audio streams in main encode. Valid only for transport stream. Must be between 0x10 and 0x1fff. Must be unique among all component streams of transport stream.	Check Mux Registry table (see Appendix A).
-490	Invalid video PID for video stream in main encode. Valid only if this is a transport stream. Must be between 0x10 and 0x1fff. Must be unique among all component streams of transport stream.	Check Mux Registry table (see Appendix A).
-491	Invalid setting for "Adjust GOP Time code" flag. Must be off (0) or on (1). Can be turned on only if VTR-control is turned on.	Check Mux Registry table (see Appendix A).
-492	Invalid mux rate supplied for main encode. Used only for transport stream. Must be between 512000 and 50,000,000.	Check Mux Registry table (see Appendix A).
-493	Invalid closed caption flag setting for main encode. Must be off (0) or on (1).	Check Mux Registry table (see Appendix A).
-494	Invalid closed caption format. Must be between 0 and 3. Meaningful only if closed caption flag is set to 1.	Check Mux Registry table (see Appendix A).

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-495	Invalid SourceEnabled setting. Must be 0 to turn OFF VTR Control, or 1 to turn it on. Can be turned on only if VTR is marked as installed in Encoder Config Registry.	Check VTR Registry table (see Appendix A).
-496	Invalid Com-port setting for VTR component. Must be 1 or 2 and must represent serial port through which encoder communicates with VTR.	Check VTR Registry table (see Appendix A).
-497	Invalid VTR adjustment for VTR component. Must be between -20 frames and + 20 frames.	Check VTR Registry table (see Appendix A).
-498	Invalid pre-roll for VTR. Must be >= 0 if VTR control is enabled. Represents number of frames earlier (-) or later (+) to start encode.	Check VTR Registry table (see Appendix A).
-499	Invalid drop frame setting. Must be 0 or 1 if this is NTSC content, or 0 if it is PAL. This value will be overridden during the encode with the actual drop-frame setting of the tape once the encode is cued and/or started.	Check VTR Registry table (see Appendix A).
-500	Invalid segment count. You must define at least one and no more than 3 durations. If VTR-control is enabled, the duration is represented by a mark-in and mark-out pair. If VTR-control is disabled, the duration is represented by the Duration time code. All of these are defined in the VTR Registry table.	Check VTR Registry table (see Appendix A).
-501	Invalid Mark-in time code (used only when VTR-Control is turned on).	Check VTR Registry table (see Appendix A).
-502	Invalid Mark-out time code (used only when VTR-Control is turned on).	Check VTR Registry table (see Appendix A).
-503	Invalid duration (used only when VTR-control is turned OFF).	Check VTR Registry table (see Appendix A).

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-504	Invalid setting of mux-file-enabled flag (which determines if a muxed file is to be stored during a system, program, or transport stream encode), or of the video-file-enabled flag or audio-file-enabled flag (which determines if the video or audio file is to be stored for an elementary-stream encode). The mux file cannot be enabled for an elementary encode, nor can the video or audio file be enabled for a muxed encode.	Check FilterMgr Registry table (see Appendix A).
-505	Invalid mux file name for main MPEG_2 file. A file name must be supplied for the stored file if the mux-file-enabled flag is turned on.	Check FilterMgr Registry table (see Appendix A).
-506	Invalid video file name supplied for an elementary-stream encode where video-file-enabled flag is turned on.	Check FilterMgr Registry table (see Appendix A).
-507	Invalid audio file name supplied for an elementary-stream encode where audio-file-enabled flag is turned on.	Check FilterMgr Registry table (see Appendix A).
-508	No file store selected. Either the mux-file-enabled or the video-file-enabled flag must be selected.	Check FilterMgr Registry table (see Appendix A).
-509	Invalid playback-enabled flag. It must be set to 0 or 1.	Check FilterMgr Registry table (see Appendix A).
-510	One of the dual-encode selections was turned on when the encode type selected was elementary stream, or when playback was turned off.	Check FilterMgr Registry table (see Appendix A).
-512	Mux database error when using EDL Editor. Unable to open the mux database table.	Could be an ODBC error. Run mdac_type.exe provided with current installation. Check database using MS Access.
-513	IBM-Video database error when using EDL Editor.	Could be an ODBC error. Run mdac_type.exe provided with current installation

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-514	IBM-Audio database error when using EDL Editor.	Could be an ODBC error. Run mdac_type.exe provided with current installation.
-515	Storage database error when using EDL Editor.	Could be an ODBC error. Run mdac_type.exe provided with current installation.
-516	VTR database error when using EDL Editor.	Could be an ODBC error. Run mdac_type.exe provided with current installation.
-517	Invalid DSN supplied when using EDL Editor.	Could be an ODBC error. Run mdac_type.exe provided with current installation. Also, check EDL Editor properties to be sure that it is associated with the correct DSN (“BroadcastArgus”) and be sure that “BroadcastArgus” is registered on the system as a database source.
-518	Attempted to access the ODBC load or save when not in EDL Editor mode.	
-519	Attempted to schedule pause/resume when dual-encoding turned on. Not allowed.	
-520	Attempted to cue WMF component when it was already cued.	State problem. May need to restart application.
-521	Attempted to Start WMF when it was already playing.	State problem. May need to restart application.
-527	The call to Stop WMF failed.	Component may be “hung up.” Try resetting. If that fails, application may need to be restarted.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-528	An attempt was made to reset the WMF component when it was not installed.	See if the Registry shows the WMF component turned ON even though WMF is not installed or licensed on this system.
-532	Video Bit Rate setting for Windows Media Format Component is out of range.	The number specified for video bit rate in the WMF Registry key is not valid.
-533	Filename setting for Windows Media Format Component is not valid.	Filename must be set to something other than NULL when attempting a WMF Encode.
-534	Audio Profile Index setting for Windows Media Format Component is out of range.	The Audio profile index set in the WMF Registry key is invalid.
-535	Video Frame Rate setting for Windows Media Format Component is out of range.	The Video Frame Rate in the WMF Registry key is invalid.
-536	Video codec setting for Windows Media Format Component is out of range.	The Video codec Setting in the WMF Registry key is invalid.
-537	Video Quality setting for Windows Media Format Component is out of range.	The Video Quality Setting in the WMF Registry key is invalid.
-540	There was an error creating an instance of the VelaAsfWriter COM object.	Verify that the WMF Argus Spectrum Component has been installed correctly.
-541	Error marshalling data into WMF thread (for callbacks).	
-548	There was an error setting up the Input profile.	The settings used to configure the input stream is invalid.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-549	There was an error setting up the Audio Input profile.	The settings used to configure the Audio input stream is invalid.
-550	There was an error setting up the Video Input profile.	The settings used to configure the Video input stream is invalid.
-552	There was an error setting the Output profile in the writer.	The video settings given may be invalid or the Windows Media Format components are not installed correctly.
-553	There was an error trying to set the filename of the WMF file.	Verify the filename given is valid.
-554 … -561	There was an error while setting up the Audio Profile for output.	Verify that the Audio Profile Index in the WMF Registry key is valid. Verify Windows Media Format Component has been installed correctly.
-562 … -571	There was an error while setting up the Audio Profile for output.	Verify that the Video settings in the WMF Registry key are valid. Verify Windows Media Format Component has been installed correctly.
-575	There was an error trying to create a Windows Media Writer object.	Verify Windows Media components have been installed correctly.
-576 … -578	There was an error trying to configure the Windows Media Writer object.	Verify Windows Media components have been installed correctly.
-582, -583	Error opening the WMF Registry.	Use RegEdit and/or CVPro-Prism to check the existence and state of the WMF table (See Appendix A).

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-584	Error Creating Main WMF Processing thread.	
-585	A filename was not supplied in the WMF Registry.	Check the WMF Registry to be sure that a file name was supplied. (For EDL Editor, this name will be generated based on the main MPEG file name.)
-586	WMF network configuration error.	Error setting up network configuration for WMF encode.
-601	WMF security violation. Attempted to use WMF dual-stream encoding when component not registered or HASP permission not granted.	Make sure WMF installation was performed, that HASP device and driver are installed on system, and that HASP device is programmed with full WMF permission.
-602	REAL security violation. Attempted to use Real dual-stream encoding when component not registered or HASP permission denied.	Make sure Real installation was performed, that HASP device and driver are installed on system, and that HASP device is programmed with full Real permission.
-603	Ligos security violation. Attempted to use Ligos dual-stream encoding when component not registered or HASP permission denied.	Make sure Ligos installation was performed, that HASP device and driver are installed on system, and that HASP device is programmed with full Ligos permission.
-610	Argus VTR configuration error. Attempted to turn on VTR control when Registry indicated that VTR is not installed.	Check EncodeConfig Registry. Turn on VTRInstalled flag if you intend to use the VTR.

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-611	Unable to create mutexes for video component.	System error? Check task manager.
-612	Unable to create IBM video COM component.	Make sure IBMVideo component is registered.
-613	Exception thrown during IBM Video initialization.	
-614	Error occurred during IBM Video stop procedure.	This may result in (or may have resulted from) “hung up” encode. May need to terminate it.
-615	Error occurred with IBM-video pause command.	
-616	Error occurred with IBM-video resume command.	
-617	Unable to create mutexes for IBM Audio component.	
-618	Error occurred during IBM Audio stop.	
-619	Error occurred with IBM Audio Pause.	
-620	Error occurred with IBM Audio Resume.	
-621	Error occurred with IBM Audio Reset.	
-622	Failure communicating with VSP hardware.	Check seating of encoder board. Run diagnostics.
-623	Failure mapping VSP.	Check seating of encoder board. Run diagnostics.
-624	VSP driver “open” command failed.	Check seating of encoder board. Run diagnostics.
-625	VSP reset failed. Unable to reset encoder.	
-626	VSP unmap failed.	
-627	Unable to create mutexes for FTP component	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-628	Unable to create FTP COM object.	Make certain that the RemoteStore component is registered.
-629	Error initializing RemoteStore component.	
-630	Error resetting RemoteStore component.	
-700	Attempted to cue Real component when it was already cued.	State problem. May need to restart application.
-704	Attempted to start Real when it was already playing.	State problem. May need to restart application.
-707	The call to stop Real failed.	Component may be "hung up." Try resetting. If that fails, application may need to be restarted.
-708	An attempt was made to reset the Real component when it was not installed. or a call to reset failed.	See if the Registry shows the Real component turned ON even though Real is not installed or licensed on this system.
-710	Failure to create one of two Real mutexes.	Check open handles using Task Manager.
-711	Error marshalling data into Real thread (for callbacks).	
-712	Error marshalling data into Real thread (for callbacks).	
-713	Error opening the Real or codec Registry.	Use RegEdit and/or CVPro-Prism to check the existence and state of the Real and codec tables (See Appendix A).

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-714	Error opening the Real or codec Registry.	Use RegEdit and/or CVProPrism to check the existence and state of the Real and codec tables (See Appendix A).
-715	Error creating main Real processing thread.	Error creating main Real processing thread.
-717	Conversion of an input string from Unicode failed.	Use CVProPrism to check the input parameters for Real.
-718	Failed to create the in-pin from the CVPro to the Real component.	Ascertain that both CinProSerCom and RealEncode are registered, as well as CVProServer. Make certain that previous encode closed down cleanly.
-719	Failed to initialize in pin listed above.	
-720	Real failed to compress video frame.	May have received bad data from CVPro. Check content of tape and quality of MPEG-2 file, if possible.
-721	Real failed to compress audio frame.	May have received bad data from CVPro. Check content of tape and quality of MPEG-2 file, if possible.
-723	Real engine failed during encode.	Use CVProPrism to check the input parameters for Real.
-724	Real engine failed during setup.	Use CVProPrism to check the input parameters for Real.
-725	Real DLL directory is not registered.	

Table B-1. Filter Manager Error/Status Codes (Continued)

Filter Manager Error/Status Codes (Continued)		
Error Code	**Meaning**	**Comments**
-726	A filename was not supplied in the Real Registry.	Check the RealNetworks Registry to be sure that a file name was supplied. (For EDL Editor, this name will be generated based on the main MPEG file name.).
-727	Mux failed to start.	Usually the result of an audio or video board failure.
-728	Video input ended.	Video input stopped unexpectedly. Check for PAL/NTSC or analog/digital conflict. Run diagnostics.
-729, -730	Audio input ended.	Audio input stopped unexpectedly. Check for PAL/NTSC or analog/digital conflict. Run diagnostics
-731	Mux outpin error.	The application was unable to send the muxed stream to the output pin. Check CPU/memory usage.
-732	Invalid VBR bit rate.	The average VBR bit rate entered must be less than the maximum bit rate.
-733	VBR not installed.	The application attempted to run a VBR encode when the VBR microcode has not been installed.
-734	Invalid Mux mark-in.	The mark-in entered for the Mux time-code adjustment is invalid.
-800	User cancelled encode after cueing, but before starting.	Shut down and restart the application before attempting another encode.

Table B-1. Filter Manager Error/Status Codes (Continued)

Index